

[illegible]

```

EEEEEEEEEE XX      XX      CCCCCCCC RRRRRRRR TTTTTTTTTT NN      NN      AAAAAA MM      MM
EEEEEEEEEE XX      XX      CCCCCCCC RRRRRRRR TTTTTTTTTT NN      NN      AAAAAA MM      MM
EE          XX      XX      CC          RR      RR      TT      NN      NN      AA      AA      MMMM  MMMM
EE          XX      XX      CC          RR      RR      TT      NN      NN      AA      AA      MMMM  MMMM
EE          XX      XX      CC          RR      RR      TT      NN      NN      AA      AA      MM  MM  MM
EEEEEEEEEE      XX      XX      CC          RRRRRRRR TTT      TT      NN      NN      AA      AA      MM  MM  MM
EEEEEEEEEE      XX      XX      CC          RRRRRRRR TTT      TT      NN      NN      AA      AA      MM  MM  MM
EE          XX      XX      CC          RR      RR      TT      NN      NN      AA      AA      MM  MM  MM
EE          XX      XX      CC          RR      RR      TT      NN      NN      AA      AA      MM  MM  MM
EE          XX      XX      CC          RR      RR      TT      NN      NN      AA      AA      MM  MM  MM
EEEEEEEEEE XX      XX      CCCCCCCC RR      RR      TT      NN      NN      AA      AA      MM  MM  MM
EEEEEEEEEE XX      XX      CCCCCCCC RR      RR      TT      NN      NN      AA      AA      MM  MM  MM
    
```

```

LL          IIIIII SSSSSSSS
LL          IIIIII SSSSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SS
LL          II      SSSSSS
LL          II      SSSSSS
LL          II      SS
LL          II      SS
LL          II      SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS
    
```



```
56 0151 1 %SBTTL 'Module table of contents'
57 0152 1
58 0153 1 ! Module table of contents:
59 0154 1 !
60 0155 1 FORWARD ROUTINE
61 0156 1     rtnam_common,
62 0157 1     exch$rtnam_copy_boot,
63 0158 1     exch$rtnam_delete,
64 0159 1     rtnam_delete_action,
65 0160 1     rtnam_init : NOVALUE,
66 0161 1     rtnam_parse_cleanup : NOVALUE,
67 0162 1     rtnam_parse_next_input,
68 0163 1     exch$rtnam_rename,
69 0164 1     rtnam_rename_action
70 0165 1 ;
71 0166 1
72 0167 1 ! EXCHANGE facility routines
73 0168 1 !
74 0169 1 EXTERNAL ROUTINE
75 0170 1     exch$cmd_cli_get_integer,
76 0171 1     exch$cmd_related_file_parse,
77 0172 1     exch$cmd_parse_filespec,
78 0173 1     exch$cmd_unwind_cli_syntax,
79 0174 1     exch$copy_namb_to_filb : NOVALUE,
80 0175 1     exch$io_rt11_read,
81 0176 1     exch$io_rt11_write,
82 0177 1     exch$moun_implied_mount,
83 0178 1     exch$rt11_create_file,
84 0179 1     exch$rt11_dircache_start : NOVALUE,
85 0180 1     exch$rt11_dircache_stop : NOVALUE,
86 0181 1     exch$rt11_dirseg_put,
87 0182 1     exch$rtacp_consolidate,
88 0183 1     exch$rtacp_find_file,
89 0184 1     exch$rt11_open_file,
90 0185 1     exch$rt11_write_cleanup : NOVALUE,
91 0186 1     exch$rt11_write_prepare : NOVALUE,
92 0187 1     exch$util_fao_buffer,
93 0188 1     exch$util_filb_allocate,
94 0189 1     exch$util_filb_release : NOVALUE,
95 0190 1     exch$util_file_error,
96 0191 1     exch$util_namb_release : NOVALUE,
97 0192 1     exch$util_radix50_from_ascii,
98 0193 1     exch$util_rmsb_allocate,
99 0194 1     exch$util_rmsb_release : NOVALUE,
100 0195 1     exch$util_rt11ctx_allocate,
101 0196 1     exch$util_rt11ctx_release : NOVALUE,
102 0197 1     exch$util_vm_allocate
103 0198 1 ;
104 0199 1
105 0200 1 ! Equated symbols:
106 0201 1 !
107 0202 1 ! LITERAL
108 0203 1 !
109 0204 1 !
110 0205 1 ! Bound declarations:
111 0206 1 !
112 0207 1 ! BIND
```

```
! Common action routine for delete and rename verbs
! Write a boot routine on an RT-11 volume
! Main entry routine for delete verb
! Secondary action routine for delete verb
! Inits common to rtnam and TYPE
! Release structures and clean up after parse
! Fetch and expand next input parameter
! Main entry routine for RENAME verb
! Secondary action routine for RENAME verb
```

```
! Get an integer value
! Parse a file specification with a related file name
! Parse a file specification
! Return on undefined qualifiers
! Copy fields from namb to the filb
! Read blocks from RT-11 volume
! Write blocks to an RT-11 volume
! Do a default mount
! Create and connect to an RT11 file
! Start directory write caching
! Finish write caching and flush directory
! Write a directory segment
! Compress unnecessary entries
! Find an RT11 file
! Connect an RT11 file
! Complete writing to an RT-11 volume
! Prepare to write to an RT-11 volume
! Format an fao string
! Allocate file context block
! Release file context block
! Tell about an rms error
! Release name block
! Convert ascii to radix50
! Allocate Files-11 control block
! Release Files-11 block
! Allocate RT-11 context block
! Release RT-11 block
! Allocate virtual memory
```


EXCH\$RTNAM
V04-000

RT-11 name manipulation routines
Module table of contents

; 113

0208 1 ! ;

M 7
16-Sep-1984 01:21:55
14-Sep-1984 12:29:08

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCRTNAM.B32;1

Page 3
(2)

EXC
V04

```
115 0209 1 GLOBAL ROUTINE rtnam_common = %SBTTL 'rtnam_common'
116 0210 2 BEGIN
117 0211 2 ++
118 0212 2
119 0213 2 FUNCTIONAL DESCRIPTION:
120 0214 2
121 0215 2 Common action routine for the DELETE and RENAME verbs for RT11 only
122 0216 2
123 0217 2 INPUTS:
124 0218 2
125 0219 2 none
126 0220 2
127 0221 2 IMPLICIT INPUTS:
128 0222 2
129 0223 2 Command parameters and qualifiers as returned from CLI$ routines. Global environment ref'd by exch$
130 0224 2
131 0225 2 OUTPUTS:
132 0226 2
133 0227 2 none
134 0228 2
135 0229 2 IMPLICIT OUTPUTS:
136 0230 2
137 0231 2 none
138 0232 2
139 0233 2 ROUTINE VALUE:
140 0234 2
141 0235 2 Success or worst error encountered.
142 0236 2
143 0237 2 SIDE EFFECTS:
144 0238 2
145 0239 2 Files may be created.
146 0240 2 --
147 0241 2
148 0242 2 $dbgtrc_prefix ('rtnam_common> ');
149 0243 2
150 0244 2 LOCAL
151 0245 2 abort,
152 0246 2 nosys_signalled,
153 0247 2 status
154 0248 2 ;
155 0249 2
156 0250 2 BIND
157 0251 2 rtnam = exch$a_gbl [excg$a_rtnam_work] : $ref_bblock ! Pointer to work area
158 0252 2 ;
159 0253 2
160 0254 2 ! Init the name used for the input file default
161 0255 2 !
162 0256 2 str$copy_dx (rtnam [rtnam$q_input_sticky_name], %ASCID '');
```



```
: 164      0257 2 ! Loop through the list of input file specifications. Errors will be signalled.
: 165      0258 2 !
: 166      0259 2 abort = false;
: 167      0260 2 nosys_signalled = false;
: 168      0261 2
: 169      0262 2 WHILE (status = rtnam_parse_next_input ()) ! Get next input file parameter
: 170      0263 2 DO
: 171      0264 2     BEGIN
: 172      0265 2     LOCAL
: 173      0266 2         nam_len,
: 174      0267 2         typ_len,
: 175      0268 2         tot_len
: 176      0269 2     ;
: 177      0270 2     BIND
: 178      0271 2         filb = rtnam [rtnam$a_inp_filb] : $ref_bblock,
: 179      0272 2         namb = filb [filb$a_assoc_namb] : $ref_bblock,
: 180      0273 2         volb = filb [filb$a_assoc_volb] : $ref_bblock,
: 181      0274 2         ctx = filb [filb$a_context] : $ref_bblock,
: 182      0275 2         nam_nam = namb [namb$q_name] : $desc_block,
: 183      0276 2         nam_typ = namb [namb$q_type] : $desc_block
: 184      0277 2     ;
: 185      0278 2
: 186      0279 2     $block_check (2, .filb, filb, 547);
: 187      0280 2     $block_check (2, .namb, namb, 548);
: 188      0281 2     $block_check (2, .volb, volb, 549);
: 189      0282 2     $block_check (2, .ctx, rt11ctx, 550);
: 190      0283 2     $logic_check (3, (.ctx [rt11ctx$a_assoc_filb] EQL .filb), 206);
: 191      0284 2     $logic_check (3, (.ctx [rt11ctx$a_assoc_volb] EQL .volb), 207);
: 192      0285 2
: 193      0286 2     ! Create the result file name in the filb
: 194      0287 2     !
: 195      0288 2     nam_len = .nam_nam [dsc$a_length];
: 196      0289 2     typ_len = .nam_typ [dsc$a_length];
: 197      0290 2     tot_len = .nam_len + .typ_len;
: 198      0291 2     ! Final length of both
: 199      0292 2     filb [filb$l_result_name_len] = .volb [volb$l_vol_ident_len] + .tot_len;
: 200      0293 2     ! Length of volume ident
: 201      0294 2     $logic_check (2, (.filb [filb$l_result_name_len] [EQU filb$s_result_name], 208);
: 202      0295 2     CH$COPY (.volb [volb$l_vol_ident_len], volb [volb$t_vol_ident],
: 203      0296 2         .nam_len, .nam_nam [dsc$a_pointer], .typ_len, .nam_typ [dsc$a_pointer],
: 204      0297 2         0, filb$s_result_name, filb [filb$t_result_name]);
: 205      0298 2
: 206      0299 2     $trace_print_fao ('looking for "'AF'", .filb [filb$l_result_name_len], filb [filb$t_result_name]);
: 207      0300 2     ! Need to be able to write to the volume
: 208      0301 2     !
: 209      0302 2     IF NOT .volb [volb$v_write]
: 210      0303 2     THEN
: 211      0304 2         BEGIN
: 212      0305 2             status = (IF .rtnam [rtnam$v_delete_command] THEN exch$nodelock ELSE exch$norenlock);
: 213      0306 2             $exch_signal (.status, 2, .filb [filb$l_result_name_len], filb [filb$t_result_name]);
: 214      0307 2             abort = true;
: 215      0308 2             END
: 216      0309 2     ELSE
: 217      0310 2         BEGIN
: 218      0311 2             ! Engage directory write caching on the volume
: 219      0312 2             !
: 220      0313 2             !
```



```
221 0314 4      exch$rt11_dircache_start (.volb);
222 0315 4
223 0316 4      WHILE exch$rtacp_find_file (.ctx, filb [filb$st_result_name] + .volb [volb$l_vol_ident_len], .tot_len
224 0317 4      DO
225 0318 4
226 0319 4      ! Two kinds of files need special treatment, so filter them before we perform any action
227 0320 4      !
228 0321 4      IF
229 0322 4
230 0323 4      ! Files with a file type of .SYS will not be touched unless /SYSTEM has been given
231 0324 4      !
232 0325 6      (BEGIN
233 0326 6      IF .ctx [rt11ctx$w_filetype] EQL r50_sys
234 0327 6      THEN
235 0328 7      BEGIN
236 0329 7
237 0330 7      ! Give a warning message the first time we pass over a .SYS file
238 0331 7      !
239 0332 8      IF (NOT .rtnam [rtnam$v_q_system])
240 0333 7      AND
241 0334 8      (NOT .nosys_signalled)
242 0335 7      THEN
243 0336 8      BEGIN
244 0337 8      nosys_signalled = true;
245 0338 8      sexch_signal (exch$nosysact);
246 0339 7      END;
247 0340 7      .rtnam [rtnam$v_q_system] ! Block has the value of the qualifier /SYSTEM
248 0341 7      END
249 0342 6      ELSE
250 0343 6      1
251 0344 5      ! File not .SYS, file is "found"
252 0345 5      END)
253 0346 5      ! And now for the second class of file. Both blocks must return true for the file to be "found"
254 0347 5      !
255 0348 4      AND
256 0349 4
257 0350 4      ! Files with a file type of .BAD will not be touched by wildcard names
258 0351 4      !
259 0352 6      (BEGIN
260 0353 6      IF .ctx [rt11ctx$w_filetype] EQL r50_bad
261 0354 6      THEN
262 0355 7      BEGIN
263 0356 7      IF .namb [namb$v_wild_name]
264 0357 7      OR
265 0358 7      .namb [namb$v_wild_type]
266 0359 7      THEN
267 0360 7      0 ! Wildcard, block returns false, file is not "found"
268 0361 7      ELSE
269 0362 7      1 ! No wildcard, block returns true, file is "found"
270 0363 7      END
271 0364 6      ELSE
272 0365 6      1 ! File not .BAD, file is "found"
273 0366 5      END)
274 0367 5
275 0368 5      ! Now the THEN clause, we have truly found this file, now work on it
276 0369 5      !
277 0370 4      THEN
```



```
278 BEGIN
279 LOCAL
280   res_buf : $bvector [filb$$result_name],           ! A buffer in which to build the act
281   res_len;
282
283   ! Create the result file name in the filb
284   !
285   nam_len = .ctx [rt11ctx$exp_fullname_len];
286   res_len = .volb [volb$vol_ident_len] + .nam_len;    ! Length of volume ident plu
287   $logic_check (2, (.res_len [EQU filb$$result_name], 125);
288   CH$COPY (.volb [volb$vol_ident_len], .volb [volb$vol_ident],      ! Volume nam
289           .nam_len, ctx [rt11ctx$exp_fullname],
290           0, filb$$result_name, res_buf);
291
292   $debug_print_fao ('file "AF" exists', .res_len, res_buf);
293
294   filb [filb$v_files_found] = true;    ! Remember that we have found a file, whether or not
295
296   ! If control/c has been hit, leave the loop now
297   IF .exch$a_gbl [excg$v_control_c]
298   THEN
299     BEGIN
300       $exch_signal ($info_stat_copy (exch$_canceled));
301       abort = true;
302       EXITLOOP;
303     END;
304
305   ! Call the appropriate secondary action routine, depending on the command
306   !
307   IF .rtnam [rtnam$v_delete_command]
308   THEN
309     status = rtnam_delete_action (.res_len, res_buf)
310   ELSE
311     BEGIN
312       status = rtnam_rename_action (.res_len, res_buf);
313       SELECTONE .status OF
314       SET
315         [exch$_parseerr, exch$_norendev, exch$_badfilename] :
316         abort = true;
317       [OTHERWISE] :
318         ;
319     TES;
320     END;
321
322   IF .abort THEN EXITLOOP;
323   END;
324
325   ! If no files were found, then scream and shout
326   !
327   IF NOT .filb [filb$v_files_found]
328   THEN
329     BEGIN
330       LOCAL
331       fao_desc : VECTOR [2, LONG];
332
333       ! Turn the expanded name into a descriptor
```

```

: 335      0428 5      !
: 336      0429 5      fao_desc [0] = .filb [filb$l_result_name_len];
: 337      0430 5      fao_desc [1] = filb [filb$t_result_name];
: 338      0431 5
: 339      0432 5      $exch_signal (exch$_filenotfound, 1, fao_desc);
: 340      0433 5
: 341      0434 4      END;
: 342      0435 4
: 343      0436 4      !??? IF .abort THEN EXITLOOP;
: 344      0437 4      END;
: 345      0438 4
: 346      0439 4      rtnam_parse_cleanup ();
: 347      0440 4      IF .abort THEN EXITLOOP;
: 348      0441 2      END;
: 349      0442 2
: 350      0443 2      RETURN .status;
: 351      0444 1      END;

```

! Release namb, clean up after parse

010E0000 00000 P.AAB:
00000000 00000 P.AAA:
00000000 00004

.TITLE EXCH\$RTNAM RT-11 name manipulation routines
.IDENT \V04-000\

.PSECT EXCH\$RTNAM_PLIT,NOWRT,2

.BLKB 0
.LONG 17694720
.ADDRESS P.AAB

.EXTRN EXCH\$CMD_CLI_GET_INTEGER
.EXTRN EXCH\$CMD_RELATED_FILE_PARSE
.EXTRN EXCH\$CMD_PARSE_FILESPEC
.EXTRN EXCH\$CMD_UNWIND_CLI_SYNTAX
.EXTRN EXCH\$COPY_NAMB_TO_FILB
.EXTRN EXCH\$IO_RT11_READ
.EXTRN EXCH\$IO_RT11_WRITE
.EXTRN EXCH\$MOON_IMPLIED_MOUNT
.EXTRN EXCH\$RT11_CREATE_FILE
.EXTRN EXCH\$RT11_DIRCACHE_START
.EXTRN EXCH\$RT11_DIRCACHE_STOP
.EXTRN EXCH\$RT11_DIRSEG_PUT
.EXTRN EXCH\$RTACP_CONSOLIDATE
.EXTRN EXCH\$RTACP_FIND_FILE
.EXTRN EXCH\$RT11_OPEN_FILE
.EXTRN EXCH\$RT11_WRITE_CLEANUP
.EXTRN EXCH\$RT11_WRITE_PREPARE
.EXTRN EXCH\$UTIL_FAO_BUFFER
.EXTRN EXCH\$UTIL_FILB_ALLOCATE
.EXTRN EXCH\$UTIL_FILB_RELEASE
.EXTRN EXCH\$UTIL_FILE_ERROR
.EXTRN EXCH\$UTIL_NAMB_RELEASE
.EXTRN EXCH\$UTIL_RADIX50_FROM_ASCII
.EXTRN EXCH\$UTIL_RMSB_ALLOCATE
.EXTRN EXCH\$UTIL_RMSB_RELEASE
.EXTRN EXCH\$UTIL_RT11CTX_ALLOCATE
.EXTRN EXCH\$UTIL_RT11CTX_RELEASE
.EXTRN EXCH\$UTIL_VM_ALLOCATE
.EXTRN EXCH\$A_GBC, STR\$COPY_DX

				OFFC 00000		
		5E	FED8	CE	9E	00002
50	00000000G	EF		18	C1	00007
			0000'	CF	9F	0000F
		58		60	D0	00013
			14	A8	9F	00016
	00000000G	00		02	FB	00019
			14	AE	7C	00020
	0000V	CF		00	FB	00023
	10	AE		50	D0	00028
		03	10	AE	E8	0002C
				0282	31	00030
		56	24	A8	D0	00033
	OC	AE	18	A6	D0	00037
50	OC	AE	00000050	8F	C1	0003C
		5A		60	9E	00045
50	OC	AE	00000058	8F	C1	00048
		59		60	9E	00051
		52	035B00FA	8F	D0	00054
		51	0223	8F	3C	0005B
		50		56	D0	00060
			00000000G	EF	16	00063
		52	010A00F7	8F	D0	00069
		51	0224	8F	3C	00070
		50	OC	AE	D0	00075
			00000000G	EF	16	00079
		57	1C	A6	D0	0007F
		52	041B00F3	8F	D0	00083
		51	0225	8F	3C	0008A
		50		57	D0	0008F
			00000000G	EF	16	00092
		5B	20	A6	D0	00098
		52	008200F4	8F	D0	0009C
		51	0226	8F	3C	000A3
		50		5B	D0	000A8
			00000000G	EF	16	000AB
	08	AE		6A	3C	000B1
	04	AE		69	3C	000B5
	08	AE	04	AE	C1	000B9
3A	6E	A7		6E	C1	000BF
	A6	8F	3A	A6	D1	000C5
	00000100			13	1B	000CD
		7E	D0	8F	9A	000CF
				01	DD	000D3
			00000000G	8F	DD	000D5
	00000000G	00		03	FB	000DB

.EXTRN	EXCH\$UTIL_BLOCK_CHECK	
.EXTRN	EXCH\$_BADLOGIC, EXCH\$_NODELLOCK	
.EXTRN	EXCH\$_NORENLOCK	
.EXTRN	EXCH\$_NOSYSACT, EXCH\$_CANCELED	
.EXTRN	EXCH\$_PARSEERR, EXCH\$_NORENDEV	
.EXTRN	EXCH\$_BADFILENAME	
.EXTRN	EXCH\$_FILENOTFOUND	
.PSECT	EXCH\$RTNAM_CODE, NOWRT, 2	
.ENTRY	RTNAM_COMMON, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	0209
	R10, RT1	
MOVAB	-296(SP), SP	
ADDL3	#24, EXCH\$A_GBL, R0	0251
PUSHAB	P.AAA	0256
MOVL	(R0), R8	
PUSHAB	20(R8)	
CALLS	#2, STR\$COPY DX	
CLRQ	NOSYS_SIGNALLED	0260
CALLS	#0, RTNAM_PARSE_NEXT_INPUT	0262
MOVL	R0, STATUS	
BLBS	STATUS, 2\$	
BRW	21\$	
MOVL	36(R8), R6	0272
MOVL	24(R6), 12(SP)	0275
ADDL3	#80, 12(SP), R0	
MOVAB	(R0), R10	
ADDL3	#88, 12(SP), R0	0276
MOVAB	(R0), R9	
MOVL	#56295674, R2	0279
MOVZWL	#547, R1	
MOVL	R6, R0	
JSB	EXCH\$UTIL_BLOCK_CHECK	
MOVL	#17432823, R2	0280
MOVZWL	#548, R1	
MOVL	12(SP), R0	
JSB	EXCH\$UTIL_BLOCK_CHECK	
MOVL	28(R6), R7	0281
MOVL	#68878579, R2	
MOVZWL	#549, R1	
MOVL	R7, R0	
JSB	EXCH\$UTIL_BLOCK_CHECK	
MOVL	32(R6), RT1	0282
MOVL	#8519924, R2	
MOVZWL	#550, R1	
MOVL	R11, R0	
JSB	EXCH\$UTIL_BLOCK_CHECK	
MOVZWL	(R10), NAM_LEN	0288
MOVZWL	(R9), TYP_LEN	0289
ADDL3	TYP_LEN, NAM_LEN, TOT_LEN	0290
ADDL3	TOT_LEN, 101(R7), 58(R6)	0291
CPL	58(R6), #256	0292
BLEQU	3\$	
MOVZBL	#208, -(SP)	
PUSHL	#1	
PUSHL	#EXCH\$_BADLOGIC	
CALLS	#3, LIB\$STOP	

20	AE	00	20	AE	0100	8F	3C	000E2	3\$:	MOVZWL	#256, 32(SP)	0293
			24	AE	5A	A6	9E	000E8		MOVAB	90(R6), 36(SP)	0295
			1C	AE	24	AE	D0	000ED		MOVL	36(SP), 28(SP)	
			69	A7	65	A7	2C	000F2		MOVCS	101(R7), 105(R7), #0, 32(SP), a28(SP)	
					1C	BE		000FA				
						2A	18	000FC		BGEQ	4\$	
			1C	AE	65	A7	C0	000FE		ADDL2	101(R7), 28(SP)	
20	AE	00	20	AE	65	A7	C2	00103		SUBL2	101(R7), 32(SP)	
			04	BA	08	AE	2C	00108		MOVCS	NAM_LEN, a4(R10), #0, 32(SP), a28(SP)	
					1C	BE		00110				
						14	18	00112		BGEQ	4\$	
			1C	AE	08	AE	C0	00114		ADDL2	NAM_LEN, 28(SP)	
20	AE	00	20	AE	08	AE	C2	00119		SUBL2	NAM_LEN, 32(SP)	
			04	B9	04	AE	2C	0011E		MOVCS	TYP_LEN, a4(R9), #0, 32(SP), a28(SP)	
					1C	BE		00126				
		30	48	A7		05	E0	00128	4\$:	BBS	#5, 72(R7), 7\$	0301
		0A	20	A8		01	E1	0012D		BBC	#1, 32(R8), 5\$	0304
			10	AE	00000000G	8F	D0	00132		MOVL	#EXCH\$_NODELLOCK, STATUS	
						08	11	0013A		BRB	6\$	
			10	AE	00000000G	8F	D0	0013C	5\$:	MOVL	#EXCH\$_NORENLOCK, STATUS	
					24	AE	DD	00144	6\$:	PUSHL	36(SP)	0305
					3A	A6	DD	00147		PUSHL	58(R6)	
						02	DD	0014A		PUSHL	#2	
					1C	AE	DD	0014C		PUSHL	STATUS	
		00000000G		00		04	FB	0014F		CALLS	#4, LIB\$SIGNAL	
		18		AE		01	D0	00156		MOVL	#1, ABORT	0306
						014C	31	0015A		BRW	20\$	0301
						57	DD	0015D	7\$:	PUSHL	R7	0314
		00000000G		EF		01	FB	0015F		CALLS	#1, EXCH\$RT11_DIRCACHE_START	
						6E	DD	00166	8\$:	PUSHL	TOT_LEN	0316
50		56			65	A7	C1	00168		ADDL3	101(R7), R6, R0	
					5A	A0	9F	0016D		PUSHAB	90(R0)	
						5B	DD	00170		PUSHL	R11	
		00000000G		EF		03	FB	00172		CALLS	#3, EXCH\$RTACP_FIND_FILE	
				03		50	E8	00179		BLBS	R0, 9\$	
						0109	31	0017C		BRW	19\$	
		7ABB		8F	3E	AB	B1	0017F	9\$:	CMPW	62(R11), #31419	0326
						1F	12	00185		BNEQ	11\$	
1A		1C		A8		03	E0	00187		BBS	#3, 28(R8), 11\$	0332
				11	14	AE	E8	0018C		BLBS	NOSYS_SIGNALLED, 10\$	0334
		14		AE		01	D0	00190		MOVL	#1, NOSYS_SIGNALLED	0337
					00000000G	8F	DD	00194		PUSHL	#EXCH\$ NOSYSACT	0338
		00000000G		00		01	FB	0019A		CALLS	#1, LIB\$SIGNAL	
				1C		03	E1	001A1	10\$:	BBC	#3, 28(R8), 8\$	0340
		OCAC		8F	3E	AB	B1	001A6	11\$:	CMPW	62(R11), #3244	0353
						1A	12	001AC		BNEQ	12\$	
50		OC		AE	0000006C	8F	C1	001AE		ADDL3	#108, 12(SP), R0	0356
AB				60		01	E0	001B7		BBS	#1, (R0), 8\$	
51		OC		AE	0000006C	8F	C1	001BB		ADDL3	#108, 12(SP), R1	0358
9E				61		02	E0	001C4		BBS	#2, (R1), 8\$	
		08		AE	46	AB	D0	001C8	12\$:	MOVL	70(R11), NAM_LEN	0378
		65		A7	08	AE	C1	001CD		ADDL3	NAM_LEN, 101(R7), RES_LEN	0379
20	AE	00000100		8F	20	AE	D1	001D4		CMPL	RES_LEN, #256	0380
						13	1B	001DC		BLEQU	13\$	
				7E	7D	8F	9A	001DE		MOVZBL	#125, -(SP)	
						01	DD	001E2		PUSHL	#1	
					00000000G	8F	DD	001E4		PUSHL	#EXCH\$_BADLOGIC	

EXCH\$RTNAM
V04-000

RT-11 name manipulation routines
rtnam_common

H 8
16-Sep-1984 01:21:55
14-Sep-1984 12:29:08

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCRTNAM.B32;1

Page 11
(4)

SA	00	69	00	0100	03	FB	001EA	13\$:	CALLS	#3, LIB\$STOP	0382
			5A	28	8F	3C	001F1		MOVZWL	#256, R10	
			59	65	AE	9E	001F6		MOVAB	RES_BUF, R9	
			A7		A7	2C	001FA		MOVCS	101(R7), 105(R7), #0, R10, (R9)	
					69		00201				
					10	18	00202		BGEQ	14\$	
			59	65	A7	C0	00204		ADDL2	101(R7), R9	
SA	00	54	5A	65	A7	C2	00208		SUBL2	101(R7), R10	
			AB	08	AE	2C	0020C		MOVCS	NAM_LEN, 84(R11), #0, R10, (R9)	
		2B	A6		69		00213				
			1B	00000000G	08	88	00214	14\$:	BISB2	#8, 43(R6)	0387
			50	00000000G	FF	E9	00218		BLBC	@EXCH\$ GBL, 15\$	0391
50	03		00		8F	D0	0021F		MOVL	#EXCH\$ CANCELED, STATUS2	0394
					03	F0	00226		INSV	#3, #0, #3, STATUS2	
					50	DD	0022B		PUSHL	STATUS2	
			00		01	FB	0022D		CALLS	#1, LIB\$SIGNAL	
			18		01	D0	00234		MOVL	#1, ABORT	0395
					4E	11	00238		BRB	19\$	0393
	11	20	A8		01	E1	0023A	15\$:	BBC	#1, 32(R8), 16\$	0401
				28	AE	9F	0023F		PUSHAB	RES_BUF	0403
				24	AE	DD	00242		PUSHL	RES_LEN	
			CF		02	FB	00245		CALLS	#2, RTNAM_DELETE_ACTION	
			AE		50	D0	0024A		MOVL	R0, STATUS	
					31	11	0024E		BRB	18\$	
				28	AE	9F	00250	16\$:	PUSHAB	RES_BUF	0406
				24	AE	DD	00253		PUSHL	RES_LEN	
			CF		02	FB	00256		CALLS	#2, RTNAM_RENAME_ACTION	
			AE		50	D0	0025B		MOVL	R0, STATUS	
			8F	10	AE	D1	0025F		CMPL	STATUS, #EXCH\$_PARSEERR	0409
					14	13	00267		BEQL	17\$	
			8F	10	AE	D1	00269		CMPL	STATUS, #EXCH\$_NORENDEV	
					0A	13	00271		BEQL	17\$	
			8F	10	AE	D1	00273		CMPL	STATUS, #EXCH\$_BADFILENAME	
					04	12	0027B		BNEQ	18\$	
			18		01	D0	0027D	17\$:	MOVL	#1, ABORT	0410
			03	18	AE	E8	00281	18\$:	BLBS	ABORT, 19\$	0416
					FE	31	00285		BRW	8\$	
					03	E0	00288	19\$:	BBS	#3, 43(R6), 20\$	0421
			2B		A6	D0	0028D		MOVL	58(R6), FAO_DESC	0429
			F8		AE	D0	00292		MOVL	36(SP), FAO_DESC+4	0430
			FC		AD	9F	00297		PUSHAB	FAO_DESC	0432
					01	DD	0029A		PUSHL	#1	
					8F	DD	0029C		PUSHL	#EXCH\$ FILENOTFOUND	
			00		03	FB	002A2		CALLS	#3, LIB\$SIGNAL	
			CF		00	FB	002A9	20\$:	CALLS	#0, RTNAM_PARSE_CLEANUP	0439
			03	18	AE	E8	002AE		BLBS	ABORT, 21\$	0440
					FD	31	002B2		BRW	1\$	
			50	10	AE	D0	002B5	21\$:	MOVL	STATUS, R0	0443
					04		002B9		RET		0444

; Routine Size: 698 bytes, Routine Base: EXCH\$RTNAM_CODE + 0000

```

: 353 0445 1 GLOBAL ROUTINE exch$rtnam_copy_boot = %SBTTL 'exch$rtnam_copy_boot'
: 354 0446 2 BEGIN
: 355 0447 2 ++
: 356 0448 2
: 357 0449 2 FUNCTIONAL DESCRIPTION:
: 358 0450 2
: 359 0451 2 Write a boot block on the RT-11 volume.
: 360 0452 2
: 361 0453 2 INPUTS:
: 362 0454 2
: 363 0455 2 none
: 364 0456 2
: 365 0457 2 IMPLICIT INPUTS:
: 366 0458 2
: 367 0459 2 Command parameters and qualifiers as returned from CLIS routines. Global environment ref'd by exch$
: 368 0460 2
: 369 0461 2 OUTPUTS:
: 370 0462 2
: 371 0463 2 none
: 372 0464 2
: 373 0465 2 IMPLICIT OUTPUTS:
: 374 0466 2
: 375 0467 2 none
: 376 0468 2
: 377 0469 2 ROUTINE VALUE:
: 378 0470 2
: 379 0471 2 Success or worst error encountered.
: 380 0472 2
: 381 0473 2 SIDE EFFECTS:
: 382 0474 2
: 383 0475 2 Block 0 and 2-5 of the target volume will be written
: 384 0476 2 !--
: 385 0477 2
: 386 0478 2 $dbgtrc_prefix ('rtnam_copy_boot> ');
: 387 0479 2
: 388 0480 2 LOCAL
: 389 0481 2 check,
: 390 0482 2 buf : $bvector [512*5], ! five block buffer for monitor file blocks
: 391 0483 2 status
: 392 0484 2 ;
: 393 0485 2
: 394 0486 2 BIND
: 395 0487 2 rtnam = exch$a_gbl [excg$a_rtnam_work] : $ref_bblock ! Pointer to work area
: 396 0488 2 ;
: 397 0489 2
: 398 0490 2 ! Allocate and/or initialize the work area
: 399 0491 2
: 400 0492 2 rtnam_init ();
```



```
: 402      0493 2 ! Get the monitor file specification. Errors will be signalled.
: 403      0494 2
: 404      0495 2 str$copy_dx (rtnam [rtnam$a_input_sticky_name], %ASCII '.SYS'); ! Default file type to .SYS
: 405      0496 2 IF (status = rtnam_parse_next_input ())
: 406      0497 2 THEN
: 407      0498 2     BEGIN
: 408      0499 2     LOCAL
: 409      0500 2         out_file : $desc_block,
: 410      0501 2         out_namb : $ref_bblock,
: 411      0502 2         nam_len,
: 412      0503 2         typ_len,
: 413      0504 2         tot_len
: 414      0505 2     ;
: 415      0506 2
: 416      0507 2     BIND
: 417      0508 2         filb = rtnam [rtnam$a_inp_filb] : $ref_bblock,
: 418      0509 2         namb = filb [filb$a_assoc_namb] : $ref_bblock,
: 419      0510 2         volb = filb [filb$a_assoc_volb] : $ref_bblock,
: 420      0511 2         ctx = filb [filb$a_context] : $ref_bblock,
: 421      0512 2         nam_nam = namb [namb$a_name] : $desc_block,
: 422      0513 2         nam_typ = namb [namb$a_type] : $desc_block
: 423      0514 2     ;
: 424      0515 2
: 425      0516 2     $block_check (2, .filb, filb, 638);
: 426      0517 2     $block_check (2, .namb, namb, 639);
: 427      0518 2     $block_check (2, .volb, volb, 640);
: 428      0519 2     $block_check (2, .ctx, rt11ctx, 641);
: 429      0520 2     $logic_check (3, (.ctx [rt11ctx$a_assoc_filb] EQL .filb), 273);
: 430      0521 2     $logic_check (3, (.ctx [rt11ctx$a_assoc_volb] EQL .volb), 274);
: 431      0522 2
: 432      0523 2     ! See if the optional output parameter is present. If so, verify that it refers to the same volb as
: 433      0524 2     ! the input. This lets us keep the syntax for COPY /BOOT compatible with RT-11.
: 434      0525 2
: 435      0526 2     $dyn_str_desc_init (out_file);
: 436      0527 2     IF (status = exch$cmd_parse_filespec (%ASCII 'OUTPUT', 0, 0, out_file, out_namb))
: 437      0528 2     THEN
: 438      0529 2         BEGIN
: 439      0530 2         REGISTER
: 440      0531 2             ignore,
: 441      0532 2             out_volb;
: 442      0533 2         out_volb = .out_namb [namb$a_assoc_volb];
: 443      0534 2         ignore = (0 NEQ (.out_namb [namb$l_nameflags] AND (namb$m_explicit_node OR namb$m_explicit_directory
: 444      0535 2             OR namb$m_explicit_name OR namb$m_explicit_type OR namb$m_explicit_version
: 445      0536 2         exch$util_namb_release (.out_namb);
: 446      0537 2         IF .out_volb NEQ .volb
: 447      0538 2         THEN
: 448      0539 2             BEGIN
: 449      0540 2                 status = exch$_nocopyboot;
: 450      0541 2                 $exch_signal (.status, 0, exch$_notsamedev);
: 451      0542 2                 rtnam_parse_cleanup ();
: 452      0543 2                 RETURN .status;
: 453      0544 2             END;
: 454      0545 2         IF .ignore
: 455      0546 2         THEN
: 456      0547 2             $exch_signal (exch$_devonly, 1, out_file);
: 457      0548 2         END;
: 458      0549 2     ;
```

```
459      0550      3      ! Create the result file name in the filb, concatenate name and type together
460      0551      3      !
461      0552      3      nam_len = .nam_nam [dsc$w_length];
462      0553      3      typ_len = .nam_typ [dsc$w_length];
463      0554      3      tot_len = .nam_len + .typ_len;
464      0555      3      ! Final length of both
465      0556      3      filb [filb$l_result_name_len] = .volb [volb$l_vol_ident_len] + .tot_len;
466      0557      3      ! Length of volume ident
467      0558      3      $logic_check (2, (.filb [filb$l_result_name_len] [EQU filb$s_result_name], 289);
468      0559      3      CH$COPY (.volb [volb$l_vol_ident_len], .volb [volb$t_vol_ident],
469      0560      3      ! Volume name
470      0561      3      .nam_len, .nam_nam [dsc$a_pointer], .typ_len, .nam_typ [dsc$a_pointer],
471      0562      3      0, filb$s_result_name, filb [filb$t_result_name]);
472      0563      3      $trace_print_fao ('looking for "!AF"', .filb [filb$l_result_name_len], filb [filb$t_result_name]);
473      0564      3      ! Need to be able to write to the volume
474      0565      3      IF NOT .volb [volb$v_write]
475      0566      3      THEN
476      0567      4      BEGIN
477      0568      4      status = exch$nocopyboot;
478      0569      4      $exch_signal (.status, 0, $warning_stat_copy (exch$writelock),
479      0570      4      2, .volb [volb$l_vol_ident_len], volb [volb$t_vol_ident]);
480      0571      4      END
481      0572      4      ! We can write, now see if we can find the monitor file
482      0573      4      !
483      0574      4      ELSE
484      0575      3      BEGIN
485      0576      4      IF exch$rtacp_find_file (.ctx, filb [filb$t_result_name] + .volb [volb$l_vol_ident_len], .tot_len)
486      0577      4      THEN
487      0578      4      BEGIN
488      0579      4      ! Remember the status of the read-check bit, then force read-checking on
489      0580      5      !
490      0581      5      check = .volb [volb$v_read_check];
491      0582      5      volb [volb$v_read_check] = true;
492      0583      5      ! Now we can do what we came here to do, read the first 5 blocks of the monitor file
493      0584      5      !
494      0585      5      $logic_check (0, (.ctx$k_buffer_blocks GEQU 5), 314);
495      0586      5      ! Need a big enough buffer for read-
496      0587      5      assumption 314 verified during compilation
497      0588      5      status = exch$io_rt11_read (.volb, .ctx [rt11ctx$l_start_block], 5, buf [0]);
498      0589      5      volb [volb$v_read_check] = .check; ! Restore check state
499      0590      5      IF NOT .status
500      0591      5      THEN
501      0592      5      RETURN .status;
502      0593      5      ! Remember the status of the write-check bit, then force write-checking on
503      0594      5      !
504      0595      5      check = .volb [volb$v_write_check];
505      0596      5      volb [volb$v_write_check] = true;
506      0597      5      ! Write the first block of the monitor file to block 0 of the volume
507      0598      5      !
508      0599      5      IF NOT (status = exch$io_rt11_write (.volb, 0, 1, buf [0]))
509      0600      5      THEN
510      0601      5      BEGIN
511      0602      5      !
512      0603      6      !
513      0604      5      !
514      0605      6      !
```



```

515      0606 6      volb [volb$v_write_check] = .check;      ! Restore the write-checking state
516      0607      RETURN .status;
517      0608      END;
518      0609
519      0610      ! Write the next four blocks of the monitor file to blocks 2 to 5 of the volume
520      0611      !
521      0612      status = exch$io_rt11_write (.volb, 2, 4, buf [512]);
522      0613      volb [volb$v_write_check] = .check; ! Restore the write-checking state
523      0614      IF NOT .status
524      0615      THEN
525      0616      RETURN .status;
526      0617
527      0618      ! Log the action if requested
528      0619      !
529      0620      IF .rtnam [rtnam$v_q_log]
530      0621      THEN
531      0622      P $exch_signal (exch$copyboot, 4, .ctx [rt11ctx$l_exp_fullname_len], ctx [rt11ctx$t_exp_fulln
532      0623      .volb [volb$_vol_ident_len], volb [volb$t_vol_ident]);
533      0624      END
534      0625
535      0626      ! The monitor file was not found, scream and shout
536      0627      !
537      0628      ELSE
538      0629      BEGIN
539      0630      LOCAL
540      0631      fao_desc : VECTOR [2, LONG];
541      0632
542      0633      ! Turn the expanded name into a descriptor
543      0634      !
544      0635      fao_desc [0] = .filb [filb$l_result_name_len];
545      0636      fao_desc [1] = filb [filb$t_result_name];
546      0637
547      0638      $exch_signal (exch$_filenotfound, 1, fao_desc);
548      0639
549      0640      END;
550      0641      END;
551      0642
552      0643      rtnam_parse_cleanup ();      ! Release namb, clean up after parse
553      0644      END;
554      0645
555      0646      RETURN .status;
556      0647      END;

```

```

00 00 54 55 50 54 55 4F 00014 P.AAF: .ASCII \OUTPUT\<0><0>
010E0006 0001C P.AAE: .LONG 17694726
00000000 00020 .ADDRESS P.AAF

.PSECT EXCH$RTNAM_PLIT,NOWRT,2

.EXTRN EXCH$GQ_DYN_STR_TEMPLATE
.EXTRN EXCH$_NOCOPYBOOT
.EXTRN EXCH$_NOTSAMEDEV

```

				OFFC 00000			
52	00000000G	5E	F5D0	CE	9E	00002	
	0000V	EF		18	C1	00007	
		CF		00	FB	0000F	
			0000'	CF	9F	00014	
50	08	AE		62	DO	00018	
	08	AE		14	C1	0001C	
				50	DD	00021	
	00000000G	00		02	FB	00023	
	0000V	CF		0C	FB	0002A	
		6E		50	DO	0002F	
		03		6E	58	00032	
			024B	31	00035		
50	04	AE		24	C1	00038	1\$:
		56		60	DO	0003D	
5A	18	A6	00000050	8F	C1	00040	
59	18	A6	00000058	8F	C1	00049	
		52	035B00FA	8F	DO	00052	
		51	027E	8F	3C	00059	
		50		56	DO	0005E	
			00000000G	EF	16	00061	
		52	010A00F7	8F	DO	00067	
		51	027F	8F	3C	0006E	
		50	18	A6	DO	00073	
			00000000G	EF	16	00077	
		57	1C	A6	DO	0007D	
		52	041B00F3	8F	DO	00081	
		51	0280	8F	3C	00088	
		50		57	DO	0008D	
			00000000G	EF	16	00090	
		5B	20	A6	DO	00096	
		52	008200F4	8F	DO	0009A	
		51	0281	8F	3C	000A1	
		50		5B	DO	000A6	
			00000000G	EF	16	000A9	
28	AE	00000000G		EF	7D	000AF	
			1C	AE	9F	000B7	
			2C	AE	9F	000BA	
				7E	7C	000BD	
			0000'	CF	9F	000BF	
	00000000G	EF		05	FB	000C3	
		6E		50	DO	000CA	
		4F		6E	E9	000CD	
		50	1C	AE	DO	000D0	
		53	74	A0	DO	000D4	
				51	D4	000D8	
0F40	8F		6C	A0	B3	000DA	
				02	13	000E0	
		52		51	D6	000E2	2\$:
				51	DO	000E4	
				50	DD	000E7	

.EXTRN	EXCH\$_DEVONLY, EXCH\$_WRITELOCK	
.EXTRN	EXCH\$_COPYBOOT	
.PSECT	EXCH\$RTNAM_CODE, NOWRT, 2	
.ENTRY	EXCH\$RTNAM_COPY_BOOT, Save R2,R3,R4,R5,R6,-	0445
MOVAB	-2608(SP), SP	
ADDL3	#24, EXCH\$A_GBL, R2	0487
CALLS	#0, RTNAM_INIT	0492
PUSHAB	P.AAC	0495
MOVL	(R2), 8(SP)	
ADDL3	#20, 8(SP), R0	
PUSHL	R0	
CALLS	#2, STR\$COPY_DX	
CALLS	#0, RTNAM_PARSE_NEXT_INPUT	0496
MOVL	R0, STATUS	
BLBS	STATUS, 1\$	
BRW	15\$	
ADDL3	#36, 4(SP), R0	0509
MOVL	(R0), R6	
ADDL3	#80, 24(R6), R10	0512
ADDL3	#88, 24(R6), R9	0513
MOVL	#56295674, R2	0516
MOVZWL	#638, R1	
MOVL	R6, R0	
JSB	EXCH\$UTIL_BLOCK_CHECK	
MOVL	#17432823, R2	0517
MOVZWL	#639, R1	
MOVL	24(R6), R0	
JSB	EXCH\$UTIL_BLOCK_CHECK	
MOVL	28(R6), R7	0518
MOVL	#68878579, R2	
MOVZWL	#640, R1	
MOVL	R7, R0	
JSB	EXCH\$UTIL_BLOCK_CHECK	
MOVL	32(R6), RT1	0519
MOVL	#8519924, R2	
MOVZWL	#641, R1	
MOVL	R11, R0	
JSB	EXCH\$UTIL_BLOCK_CHECK	
MOVQ	TMPL, DESC	0526
PUSHAB	OUT_NAMB	0527
PUSHAB	OUT_FILE	
CLRQ	-(SP)	
PUSHAB	P.AAE	
CALLS	#5, EXCH\$CMD_PARSE_FILESPEC	
MOVL	R0, STATUS	
BLBC	STATUS, 4\$	
MOVL	OUT_NAMB, R0	0533
MOVL	116(R0), OUT_VOLB	
CLRL	R1	0534
BITW	108(R0), #3904	
BEQL	2\$	
INCL	R1	
MOVL	R1, IGNORE	
PUSHL	R0	0536

		00000000G	EF		01	FB	000E9	CALLS	#1, EXCH\$UTIL_NAMB_RELEASE		
			57		53	D1	000F0	CMPL	OUT_VOLB, R7		0537
					15	13	000F3	BEQL	3\$		
		6E	00000000G		8F	D0	000F5	MOVL	#EXCH\$_NOCOPYBOOT, STATUS		0540
			00000000G		8F	DD	000FC	PUSHL	#EXCH\$_NOTSAMEDEV		0541
					7E	D4	00102	CLRL	-(SP)		
				08	AE	DD	00104	PUSHL	STATUS		
					016D	31	00107	BRW	13\$		
				12	52	E9	0010A	3\$: BLBC	IGNORE, 4\$		0545
					28	AE	9F	0010D	PUSHAB	OUT_FILE	0547
					01	DD	00110	PUSHL	#1		
			00000000G		8F	DD	00112	PUSHL	#EXCH\$_DEVONLY		
		00000000G	00		03	FB	00118	CALLS	#3, LIB\$SIGNAL		0552
			18	AE	6A	3C	0011F	4\$: MOVZWL	(R10), NAM_LEN		0553
			10	AE	69	3C	00123	MOVZWL	(R9), TYP_LEN		0554
			18	AE	10	AE	C1	00127	ADDL3	TYP_LEN, NAM_LEN, TOT_LEN	
14	AE		65	A7	14	AE	C1	0012E	ADDL3	TOT_LEN, 101(R7), 58(R6)	0555
3A	A6		00000100	8F	3A	A6	D1	00135	CMPL	58(R6), #256	0556
					14	1B	0013D	BLEQU	5\$		
				7E	0121	8F	3C	0013F	MOVZWL	#289, -(SP)	
					01	DD	00144	PUSHL	#1		
			00000000G		8F	DD	00146	PUSHL	#EXCH\$_BADLOGIC		
			00		03	FB	0014C	CALLS	#3, LIB\$STOP		
			08	AE	0100	8F	3C	00153	5\$: MOVZWL	#256, 8(SP)	0557
			0C	AE	5A	A6	9E	00159	MOVAB	90(R6), 12(SP)	0559
				58	0C	AE	D0	0015E	MOVL	12(SP), R8	
08	AE		00	69	A7	2C	00162	MOVCS	101(R7), 105(R7), #0, 8(SP), (R8)		
					68		0016A				
					26	18	0016B	BGEQ	6\$		
				58	65	A7	C0	0016D	ADDL2	101(R7), R8	
				08	65	A7	C2	00171	SUBL2	101(R7), 8(SP)	
08	AE		00	04	BA	2C	00176	MOVCS	NAM_LEN, @4(R10), #0, 8(SP), (R8)		
					68		0017E				
					12	18	0017F	BGEQ	6\$		
				58	18	AE	C0	00181	ADDL2	NAM_LEN, R8	
				08	18	AE	C2	00185	SUBL2	NAM_LEN, 8(SP)	
08	AE		00	04	B9	2C	0018A	MOVCS	TYP_LEN, @4(R9), #0, 8(SP), (R8)		
					68		00192				
				52	48	A7	9E	00193	6\$: MOVAB	72(R7), R2	0565
			23	62	05	E0	00197	BBS	#5, (R2), 7\$		
				6E	00000000G	8F	D0	0019B	MOVL	#EXCH\$_NOCOPYBOOT, STATUS	0568
					69	A7	9F	001A2	PUSHAB	105(R7)	0570
					65	A7	DD	001A5	PUSHL	101(R7)	
					02	DD	001A8	PUSHL	#2		
				50	00000000G	8F	D0	001AA	MOVL	#EXCH\$_WRITELOCK, STATUS2	
				50	07	8A	001B1	BICB2	#7, STATUS2		
					50	DD	001B4	PUSHL	STATUS2		
					7E	D4	001B6	CLRL	-(SP)		
					14	AE	DD	001B8	PUSHL	STATUS	
					009B	31	001BB	BRW	11\$		
					14	AE	DD	001BE	7\$: PUSHL	TOT_LEN	0578
			50	56	65	A7	C1	001C1	ADDL3	101(R7), R6, R0	
					5A	A0	9F	001C6	PUSHAB	90(R0)	
					5B	DD	001C9	PUSHL	R11		
			00000000G	EF	03	FB	001CB	CALLS	#3, EXCH\$RTACP_FIND_FILE		
					50	EB	001D2	BLBS	R0, 8\$		
					008A	31	001D5	BRW	12\$		

EXCH\$RTNAM
V04-000

RT-11 name manipulation routines
exch\$rtnam_copy_boot

B 9
16-Sep-1984 01:21:55
14-Sep-1984 12:29:08

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCRTNAM.B32;1

Page 18
(6)

53	62	01	01	EF	001D8	8\$:	EXTZV	#1, #1, (R2), CHECK	0584	
		62	02	88	001DD		BISB2	#2, (R2)	0585	
			AE	9F	001E0		PUSHAB	BUF	0590	
			05	DD	001E3		PUSHL	#5		
			72	AB	DD	001E5	PUSHL	114(R11)		
			57	DD	001E8		PUSHL	R7		
	00000000G	EF	04	FB	001EA		CALLS	#4, EXCH\$IO_RT11_READ		
62	01	01	50	DD	001F1		MOVL	R0, STATUS	0591	
		3E	53	FO	001F4		INSV	CHECK, #1, #1, (R2)	0592	
53	62	01	6E	E9	001F9		BLBC	STATUS, 10\$	0598	
		62	02	EF	001FC		EXTZV	#2, #1, (R2), CHECK	0599	
			04	88	00201		BISB2	#4, (R2)	0603	
			30	AE	9F	00204	PUSHAB	BUF		
				01	DD	00207	PUSHL	#1		
				7E	D4	00209	CLRL	-(SP)		
				57	DD	0020B	PUSHL	R7		
	00000000G	EF	04	FB	0020D		CALLS	#4, EXCH\$IO_RT11_WRITE		
		6E	50	DD	00214		MOVL	R0, STATUS		
62	01	07	6E	E8	00217		BLBS	STATUS, 9\$		
		02	53	FO	0021A		INSV	CHECK, #2, #1, (R2)	0606	
			62	11	0021F		BRB	15\$	0607	
			0230	CE	9F	00221	9\$:	PUSHAB	BUF+512	0612
				04	DD	00225	PUSHL	#4		
				02	DD	00227	PUSHL	#2		
				57	DD	00229	PUSHL	R7		
	00000000G	EF	04	FB	0022B		CALLS	#4, EXCH\$IO_RT11_WRITE		
		6E	50	DD	00232		MOVL	R0, STATUS		
62	01	02	53	FO	00235		INSV	CHECK, #2, #1, (R2)	0613	
		46	6E	E9	0023A	10\$:	BLBC	STATUS, 15\$	0614	
	52	04	1C	C1	0023D		ADDL3	#28, 4(SP), R2	0620	
			62	E9	00242		BLBC	(R2), 14\$		
			69	A7	9F	00245	PUSHAB	105(R7)	0623	
			65	A7	DD	00248	PUSHL	101(R7)		
			54	AB	9F	0024B	PUSHAB	84(R11)		
			46	AB	DD	0024E	PUSHL	70(R11)		
				04	DD	00251	PUSHL	#4		
	00000000G	00	8F	DD	00253		PUSHL	#EXCH\$ COPYBOOT		
			06	FB	00259	11\$:	CALLS	#6, LIB\$SIGNAL		
			1C	11	00260		BRB	14\$	0578	
	20	AE	A6	DD	00262	12\$:	MOVL	58(R6), FAO_DESC	0635	
	24	AE	AE	DD	00267		MOVL	12(SP), FAO_DESC+4	0636	
			20	AE	9F	0026C	PUSHAB	FAO_DESC	0638	
				01	DD	0026F	PUSHL	#1		
	00000000G	00	8F	DD	00271		PUSHL	#EXCH\$ FILENOTFOUND		
	0000V	CF	03	FB	00277	13\$:	CALLS	#3, LIB\$SIGNAL		
		50	00	FB	0027E	14\$:	CALLS	#0, RTNAM_PARSE_CLEANUP	0643	
			6E	DD	00283	15\$:	MOVL	STATUS, R0	0646	
			04	00286			RET		0647	

; Routine Size: 647 bytes, Routine Base: EXCH\$RTNAM_CODE + 02BA


```

: 558 0648 1 GLOBAL ROUTINE exch$rtnam_delete = %SBTTL 'exch$rtnam_delete'
: 559 0649 2 BEGIN
: 560 0650 2 ++
: 561 0651 2
: 562 0652 2 FUNCTIONAL DESCRIPTION:
: 563 0653 2
: 564 0654 2 Entry routine for the DELETE verb for RT11 only
: 565 0655 2
: 566 0656 2 INPUTS:
: 567 0657 2
: 568 0658 2 none
: 569 0659 2
: 570 0660 2 IMPLICIT INPUTS:
: 571 0661 2
: 572 0662 2 Command parameters and qualifiers as returned from CLI$ routines. Global environment ref'd by exch$
: 573 0663 2
: 574 0664 2 OUTPUTS:
: 575 0665 2
: 576 0666 2 none
: 577 0667 2
: 578 0668 2 IMPLICIT OUTPUTS:
: 579 0669 2
: 580 0670 2 none
: 581 0671 2
: 582 0672 2 ROUTINE VALUE:
: 583 0673 2
: 584 0674 2 Success or worst error encountered.
: 585 0675 2
: 586 0676 2 SIDE EFFECTS:
: 587 0677 2
: 588 0678 2 Files may be deleted.
: 589 0679 2 --
: 590 0680 2
: 591 0681 2 $dbgtrc_prefix ('rtnam_delete> ');
: 592 0682 2
: 593 0683 2 LOCAL
: 594 0684 2 rtnam : $ref_bblock ! Pointer to work area
: 595 0685 2 ;
: 596 0686 2
: 597 0687 2
: 598 0688 2 ! Allocate and/or initialize the work area
: 599 0689 2
: 600 0690 2 rtnam_init ();
: 601 0691 2
: 602 0692 2 ! Get pointers that we need. Have to wait until work area allocated by init call
: 603 0693 2
: 604 0694 2 rtnam = .exch$a_gbl [excg$a_rtnam_work]; ! Pointer to work area
: 605 0695 2 rtnam [rtnam$v_delete_command] = true;
: 606 0696 2
: 607 0697 2 ! Do the rest from the common routine
: 608 0698 2
: 609 0699 2 RETURN rtnam_common ();
: 610 0700 1 END;
```

EXCH\$RTNAM
V04-000

RT-11 name manipulation routines
exch\$rtnam_delete

D 9
16-Sep-1984 01:21:55
14-Sep-1984 12:29:08

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCRTNAM.B32;1

Page 20
(7)

0000V	CF		00	FB	00002
	50	00000000G	EF	D0	00007
	50	18	A0	D0	0000E
20	A0		02	88	00012
FAA4	CF		00	FB	00016
			04	00	0001B

.ENTRY	EXCH\$RTNAM_DELETE, Save nothing
CALLS	#0, RTNAM_INIT
MOVL	EXCH\$A_GBC, R0
MOVL	24(R0), RTNAM
BISB2	#2, 32(RTNAM)
CALLS	#0, RTNAM_COMMON
RET	

:	0648
:	0690
:	0694
:	
:	0695
:	0699
:	0700

; Routine Size: 28 bytes, Routine Base: EXCH\$RTNAM_CODE + 0541


```

: 612 0701 1 GLOBAL ROUTINE rtnam_delete_action (res_len, res_buf) = %SBTTL 'rtnam_delete_action (res_len, res_buf)'
: 613 0702 2 BEGIN
: 614 0703 2 ++
: 615 0704 2
: 616 0705 2 FUNCTIONAL DESCRIPTION:
: 617 0706 2
: 618 0707 2 Secondary action routine for the DELETE verb for RT11 only
: 619 0708 2
: 620 0709 2 INPUTS:
: 621 0710 2
: 622 0711 2 res_len - length of result name
: 623 0712 2 res_buf - address of result name
: 624 0713 2
: 625 0714 2 IMPLICIT INPUTS:
: 626 0715 2
: 627 0716 2 Command parameters and qualifiers as returned from CLI$ routines. Global environment ref'd by exch$
: 628 0717 2
: 629 0718 2 OUTPUTS:
: 630 0719 2
: 631 0720 2 none
: 632 0721 2
: 633 0722 2 IMPLICIT OUTPUTS:
: 634 0723 2
: 635 0724 2 none
: 636 0725 2
: 637 0726 2 ROUTINE VALUE:
: 638 0727 2
: 639 0728 2 Success or worst error encountered.
: 640 0729 2
: 641 0730 2 SIDE EFFECTS:
: 642 0731 2
: 643 0732 2 Files may be deleted.
: 644 0733 2 --
: 645 0734 2
: 646 0735 2 $dbgtrc_prefix ('rtnam_delete_action> ');
: 647 0736 2
: 648 0737 2 LOCAL
: 649 0738 2 status
: 650 0739 2 ;
: 651 0740 2
: 652 0741 2 BIND
: 653 0742 2 rtnam = exch$a_gbl [excg$a_rtnam_work] : $ref_bblock, ! Pointer to work area
: 654 0743 2 filb = rtnam [rtnam$a_inp_filb] : $ref_bblock;
: 655 0744 2 volb = filb [filb$a_assoc_volb] : $ref_bblock;
: 656 0745 2 ctx = filb [filb$a_context] : $ref_bblock;
: 657 0746 2 ent = ctx [rt11ctx$a_ent_address] : $ref_bblock
: 658 0747 2 ;
: 659 0748 2
: 660 0749 2 $block_check (2, .filb, filb, 554);
: 661 0750 2 $block_check (2, .volb, volb, 555);
: 662 0751 2 $block_check (2, .ctx, rt11ctx, 556); ! Make sure that it is what we think it is
: 663 0752 2
: 664 0753 2 ! Verify that it is ok to delete the existing file
: 665 0754 2
: 666 0755 2 IF .ctx [rt11ctx$v_typ_protected] ! Can't delete protected files
: 667 0756 2 THEN
: 668 0757 3 $exch_signal_return (exch$_notdeleted, 2, .res_len, .res_buf, exch$_rtprotect)
```

```

669 0758 3
670 0759 ! Looks good, delete it now
671 0760
672 0761 ELSE
673 0762 BEGIN
674 0763
675 0764 $trace_print_fao ('deleting '!Af'', .res_len, .res_buf);
676 0765
677 0766 ! Mark the entry as deleted
678 0767
679 0768 ent [rt11ent$b_type_byte] = rt11ent$m_typ_empty;
680 0769
681 0770 ! Also change the type in the context buffer, since it will be used by EXCH$RTACP_CHECK_POSITION
682 0771
683 0772 ctx [rt11ctx$b_type_byte] = rt11ent$m_typ_empty;
684 0773
685 0774 exch$rtnam_dirseg_put (.volb, .ctx [rt11ctx$l_seg_number]); ! Write the directory segment
686 0775
687 0776 IF .rtnam [rtnam$v_q_log]
688 0777 THEN
689 0778 $exch_signal (exch$_deleted, 2, .res_len, .res_buf);
690 0779
691 0780 END;
692 0781
693 0782 RETURN true;
694 0783 1 END;
```

```

57 00000000G 00FC 00000
56 00000000G EF 9E 00002
50 00000000G EF 18 C1 00010
54 60 D0 00018
55 24 A4 1C C1 0001B
50 24 A4 20 C1 00020
53 60 D0 00025
52 035B00FA 8F D0 00028
51 022A 8F 3C 0002F
50 24 A4 D0 00034
66 16 00038
52 041B00F3 8F D0 0003A
51 022B 8F 3C 00041
50 65 D0 00046
66 16 00049
52 008200F4 8F D0 0004B
51 022C 8F 3C 00052
50 53 D0 00057
66 16 0005A
39 A3 95 0005C
1C 18 0005F
52 00000000G 8F D0 00061
00000000G 8F DD 00068
```

```

.EXTRN EXCH$_NOTDELETED
.EXTRN EXCH$_RTPROTECT
.EXTRN EXCH$_DELETED

.ENTRY RTNAM_DELETE_ACTION, Save R2,R3,R4,R5,R6,R7 ; 0701
MOVAB LIB$SIGNAL, R7
MOVAB EXCH$UTIL_BLOCK_CHECK, R6
ADDL3 #24, EXCH$A_GBL, R0
MOVL (R0), R4
ADDL3 #28, 36(R4), R5
ADDL3 #32, 36(R4), R0
MOVL (R0), R3
MOVL #56295674, R2
MOVZWL #554, R1
MOVL 36(R4), R0
JSB EXCH$UTIL_BLOCK_CHECK
MOVL #68878579, R2
MOVZWL #555, R1
MOVL (R5), R0
JSB EXCH$UTIL_BLOCK_CHECK
MOVL #8519924, R2
MOVZWL #556, R1
MOVL R3, R0
JSB EXCH$UTIL_BLOCK_CHECK
TSTB 57(R3)
BGEQ 1$
MOVL #EXCH$_NOTDELETED, TEMP
PUSHL #EXCH$_RTPROTECT
```

```

0742
0743
0744
0745
0746
0749
0750
0751
0755
0757
```


EXCH\$RTNAM
V04-000

RT-11 name manipulation routines
rtnam_delete_action (res_len, res_buf)

G 9
16-Sep-1984 01:21:55
14-Sep-1984 12:29:08

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCRTNAM.B32;1

Page 23
(8)

	7E	04	AC	7D	0006E	MOVQ	RES_LEN, -(SP)	:	
			02	DD	00072	PUSHL	#2	:	
			52	DD	00074	PUSHL	TEMP	:	
	67		05	FB	00076	CALLS	#5, LIB\$SIGNAL	:	
	50		52	D0	00079	MOVL	TEMP, R0	:	
				04	0007C	RET		:	
	50	7E	A3	D0	0007D	1\$:	MOVL	126(R3), R0	0768
01	A0		02	90	00081		MOVB	#2, 1(R0)	
39	A3		02	90	00085		MOVB	#2, 57(R3)	0772
		76	A3	DD	00089		PUSHL	118(R3)	0774
			65	DD	0008C		PUSHL	(R5)	
00000000G	EF		02	FB	0008E		CALLS	#2, EXCH\$RT11_DIRSEG_PUT	
	0F	1C	A4	E9	00095		BLBC	28(R4), 2\$	0776
	7E	04	AC	7D	00099		MOVQ	RES_LEN, -(SP)	0778
			02	DD	0009D		PUSHL	#2	
		00000000G	8F	DD	0009F		PUSHL	#EXCH\$ DELETED	
	67		04	FB	000A5		CALLS	#4, LIB\$SIGNAL	
	50		01	D0	000A8	2\$:	MOVL	#1, R0	0782
			04	000AB			RET		0783

; Routine Size: 172 bytes, Routine Base: EXCH\$RTNAM_CODE + 055D

```

: 696 0784 1 GLOBAL ROUTINE rtnam_init : NOVALUE = %SBTTL 'rtnam_init'
: 697 0785 2 BEGIN
: 698 0786 3 ++
: 699 0787 4
: 700 0788 5 FUNCTIONAL DESCRIPTION:
: 701 0789 6
: 702 0790 7     Common init routine for the rtnam verbs
: 703 0791 8
: 704 0792 9 INPUTS:
: 705 0793 10
: 706 0794 11     none
: 707 0795 12
: 708 0796 13 IMPLICIT INPUTS:
: 709 0797 14
: 710 0798 15     Command parameters and qualifiers as returned from CLI$ routines.  Global environment ref'd by exch$
: 711 0799 16
: 712 0800 17 OUTPUTS:
: 713 0801 18
: 714 0802 19     none
: 715 0803 20
: 716 0804 21 IMPLICIT OUTPUTS:
: 717 0805 22
: 718 0806 23     none
: 719 0807 24
: 720 0808 25 ROUTINE VALUE:
: 721 0809 26
: 722 0810 27     none
: 723 0811 28
: 724 0812 29 SIDE EFFECTS:
: 725 0813 30
: 726 0814 31     Files may be created.
: 727 0815 32 --
: 728 0816 33
: 729 0817 34 $dbgtrc_prefix ('rtnam_init> ');
: 730 0818 35
: 731 0819 36 LOCAL
: 732 0820 37     status
: 733 0821 38     ;
: 734 0822 39
: 735 0823 40 BIND
: 736 0824 41     rtnam = exch$a_gbl [excg$a_rtnam_work] : $ref_bblock      ! Pointer to work area
: 737 0825 42     ;
: 738 0826 43
: 739 0827 44 ! The COPY /BOOT command does not support /LOG or /SYSTEM.  Set up a handler to do a RETURN when we would
: 740 0828 45 ! normally get the MSG$_SYNTAX error.
: 741 0829 46
: 742 0830 47 ENABLE
: 743 0831 48     exch$cmd_unwind_cli_syntax;
: 744 0832 49
: 745 0833 50 ! If our pointer is null, we need to allocate and initialize the work area
: 746 0834 51
: 747 0835 52 IF .rtnam EQL 0
: 748 0836 53 THEN
: 749 0837 54     BEGIN
: 750 0838 55
: 751 0839 56         ! Get the right sized chunk of memory
: 752 0840 57         !

```



```

753      0841      3      rtnam = exch$util_vm_allocate (exchblk$s_rtnam);
754      0842
755      0843      ! Set the ident fields
756      0844      !
757      0845      $block_init (.rtnam, rtnam);
758      0846
759      0847      ! Set the dynamic strings
760      0848      !
761      0849      $dyn_str_desc_init (rtnam [rtnam$q_input_filename]);
762      0850      $dyn_str_desc_init (rtnam [rtnam$q_output_filename]);
763      0851      $dyn_str_desc_init (rtnam [rtnam$q_input_sticky_name]);
764      0852
765      0853      END
766      0854      ELSE
767      0855      BEGIN
768      0856
769      0857      ! Free the dynamic strings and the Chicago 7
770      0858      !
771      0859      str$free1_dx (rtnam [rtnam$q_input_filename]);
772      0860      str$free1_dx (rtnam [rtnam$q_output_filename]);
773      0861      str$free1_dx (rtnam [rtnam$q_input_sticky_name]);
774      0862
775      0863      END;
776      0864
777      0865      ! Get some confidence that our work area is valid
778      0866      !
779      0867      $block_check (2, .rtnam, rtnam, 541);
780      0868
781      0869      ! Set the last part of the block to nulls
782      0870      !
783      0871      CH$FILL (0, rtnam$k_end_zero - rtnam$k_start_zero, .rtnam + rtnam$k_start_zero);
784      0872
785      0873      ! Get the global boolean qualifiers common to all (COPY /BOOT, DELETE, RENAME) commands
786      0874      !
787      0875      rtnam [rtnam$v_q_log] = cli$present (%ASCII 'LOG'); ! global
788      0876
789      0877      ! Get global booleans common to Delete and Rename only, COPY /BOOT will unwind, so nothing that COPY/BOOT ne
790      0878      ! can be past this point
791      0879      !
792      0880      rtnam [rtnam$v_q_system] = cli$present (%ASCII 'SYSTEM'); ! global
793      0881      !\ rtnam [rtnam$v_q_confirm] = cli$present (%ASCII 'CONFIRM'); ! global
794      0882
795      0883      RETURN;
796      0884      1 END;
```

```

.PSECT EXCH$RTNAM_PLIT,NOWRT,2

00 47 4F 4C 00024 P.AAH: .ASCII \LOG\<0>
010E0003 00028 P.AAG: .LONG 17694723
00000000 0002C .ADDRESS P.AAH
00 00 4D 45 54 53 59 53 00030 P.AAJ: .ASCII \SYSTEM\<0><0>
010E0006 00038 P.AAI: .LONG 17694726
00000000 0003C .ADDRESS P.AAJ

.EXTRN STR$FREE1_DX, CLI$PRESENT
```

				.PSECT	EXCH\$RTNAM_CODE,NOWRT,2	
				.ENTRY	RTNAM_INIT, Save R2,R3,R4,R5,R6,R7,R8	0784
				MOVAB	CLIS\$PRESENT, R8	
				MOVAB	STR\$FREE1_DX, R7	
52	00000000G	EF	18	ADDL3	#24, EXCH\$A_GBL, R2	0824
		6D	CF	MOVAL	3\$, (FP)	
			62	TSTL	(R2)	0835
			44	BNEQ	1\$	
			34	PUSHL	#52	0841
	00000000G	EF	01	CALLS	#1, EXCH\$UTIL_VM_ALLOCATE	
		62	50	MOVL	R0, (R2)	
08		A0	34	MOVW	#52, 8(R0)	0845
0A		A0	0E	MNEGB	#14, 10(R0)	
		50	62	MOVL	(R2), R0	0849
		53	EF	MOVL	TMPL, R3	
		60	53	MOVL	R3, (R0)	
		51	EF	MOVL	TMPL+4, R1	
	04	A0	51	MOVL	R1, 4(R0)	
50		62	0C	ADDL3	#12, (R2), R0	0850
		60	53	MOVL	R3, (R0)	
	04	A0	51	MOVL	R1, 4(R0)	
50		62	14	ADDL3	#20, (R2), R0	0851
		60	53	MOVL	R3, (R0)	
	04	A0	51	MOVL	R1, 4(R0)	
			13	BRB	2\$	0835
			62	PUSHL	(R2)	0859
		67	01	CALLS	#1, STR\$FREE1_DX	
7E		62	0C	ADDL3	#12, (R2), -(SP)	0860
		67	01	CALLS	#1, STR\$FREE1_DX	
7E		62	14	ADDL3	#20, (R2), -(SP)	0861
		67	01	CALLS	#1, STR\$FREE1_DX	
		56	62	MOVL	(R2), R6	0867
		52	8F	MOVL	#3408114, R2	
	003400F2	51	8F	MOVZWL	#541, R1	
	021D	50	56	MOVL	R6, R0	
			EF	JSB	EXCH\$UTIL_BLOCK_CHECK	
18	00	6E	00	MOVC5	#0, (SP), #0, #24, 28(R6)	0871
			A6			
		1C	CF	PUSHAB	P.AAG	0875
		0000'	01	CALLS	#1, CLIS\$PRESENT	
1C	A6	01	50	INSV	R0, #0, #1, 28(R6)	
			CF	PUSHAB	P.AAI	0880
		0000'	01	CALLS	#1, CLIS\$PRESENT	
1C	A6	01	50	INSV	R0, #3, #1, 28(R6)	
		68	04	RET		0884
		03	0000	.WORD	Save nothing	0824
			7E	CLRL	-(SP)	
			5E	PUSHL	SP	
	04	7E	AC	MOVQ	4(AP), -(SP)	
00000000G		EF	03	CALLS	#3, EXCH\$CMD_UNWIND_CLI_SYNTAX	
			04	RET		
			04		000C3	

; Routine Size: 196 bytes, Routine Base: EXCH\$RTNAM_CODE + 0609


```

798 0885 1 GLOBAL ROUTINE rtnam_parse_cleanup : NOVALUE = %SBTTL 'rtnam_parse_cleanup'
799 0886 2 BEGIN
800 0887 2 ++
801 0888 2
802 0889 2 FUNCTIONAL DESCRIPTION:
803 0890 2
804 0891 2         Clean up after a successful parse.  Release the namb and other structures.
805 0892 2
806 0893 2 INPUTS:
807 0894 2
808 0895 2         none
809 0896 2
810 0897 2 IMPLICIT INPUTS:
811 0898 2
812 0899 2         rtnam$a_inp_namb field in rtnam work area
813 0900 2
814 0901 2 OUTPUTS:
815 0902 2
816 0903 2         none
817 0904 2
818 0905 2 IMPLICIT OUTPUTS:
819 0906 2
820 0907 2         none
821 0908 2
822 0909 2 ROUTINE VALUE:
823 0910 2
824 0911 2         none
825 0912 2
826 0913 2 SIDE EFFECTS:
827 0914 2
828 0915 2         none
829 0916 2 --
830 0917 2
831 0918 2 $dbgtrc_prefix ('rtnam_parse_cleanup> ');
832 0919 2
833 0920 2 BIND
834 0921 2     rtnam = exch$a_gbl [excg$a_rtnam_work] : $ref_bblock, ! Pointer to work area
835 0922 2     filb = rtnam [rtnam$a_inp_filb] : $ref_bblock, ! Filb for the input
836 0923 2     namb = filb [filb$a_assoc_namb] : $ref_bblock, ! Namb for the input
837 0924 2     volb = filb [filb$a_assoc_volb] : $ref_bblock, ! Volb for the input
838 0925 2     ctx = filb [filb$a_context] : $ref_bblock ! Volume specific context
839 0926 2 ;
840 0927 2
841 0928 2 $debug_print_lit ('entry');
842 0929 2 $block_check (2, .filb, filb, 543);
843 0930 2 $block_check (2, .namb, namb, 544);
844 0931 2 $block_check (2, .volb, volb, 545);
845 0932 2 $block_check (2, .ctx, rt11ctx, 546);
```

```

: 847      0933 2 ! If we have been able to write, then clean things up
: 848      0934 2 !
: 849      0935 2 IF .volb [volb$write]
: 850      0936 2 THEN
: 851      0937 2 BEGIN
: 852      0938 2
: 853      0939 2 ! Compress unnecessary entries from the directory. 0 means not to restructure the directory.
: 854      0940 2 !
: 855      0941 2     exch$rtacp_consolidate (.volb, 0);
: 856      0942 2
: 857      0943 2 ! Turn caching off and flush any modified directory segments
: 858      0944 2 !
: 859      0945 2     exch$rt11_dircache_stop (.volb);
: 860      0946 2
: 861      0947 2 END;
: 862      0948 2
: 863      0949 2 ! Release the context block
: 864      0950 2 !
: 865      0951 2     exch$util_rt11ctx_release (.ctx);
: 866      0952 2
: 867      0953 2 ! Release the input namb
: 868      0954 2 !
: 869      0955 2     exch$util_namb_release (.namb);
: 870      0956 2
: 871      0957 2 ! Release the input filb
: 872      0958 2 !
: 873      0959 2     exch$util_filb_release (.filb);
: 874      0960 2
: 875      0961 2 RETURN;
: 876      0962 1 END;
```

			00FC 00000	.ENTRY	RTNAM_PARSE_CLEANUP, Save R2,R3,R4,R5,R6,R7	: 0885
		57 00000000G	EF 9E 00002	MOVAB	EXCH\$UTIL_BLOCK_CHECK, R7	
50 00000000G		EF	18 C1 00009	ADDL3	#24, EXCH\$A_GBL, R0	: 0921
54		60	24 C1 00011	ADDL3	#36, (R0), R4	: 0922
56		64	18 C1 00015	ADDL3	#24, (R4), R6	: 0923
53		64	1C C1 00019	ADDL3	#28, (R4), R3	: 0924
55		64	20 C1 0001D	ADDL3	#32, (R4), R5	: 0925
		52 035B00FA	8F D0 00021	MOVL	#56295674, R2	: 0929
		51 021F	8F 3C 00028	MOVZWL	#543, R1	
		50	64 D0 0002D	MOVL	(R4), R0	
			67 16 00030	JSB	EXCH\$UTIL_BLOCK_CHECK	
		52 010A00F7	8F D0 00032	MOVL	#17432823, R2	: 0930
		51 0220	8F 3C 00039	MOVZWL	#544, R1	
		50	66 D0 0003E	MOVL	(R6), R0	
			67 16 00041	JSB	EXCH\$UTIL_BLOCK_CHECK	
		53	63 D0 00043	MOVL	(R3), R3	: 0931
		52 041B00F3	8F D0 00046	MOVL	#68878579, R2	
		51 0221	8F 3C 0004D	MOVZWL	#545, R1	
		50	53 D0 00052	MOVL	R3, R0	
			67 16 00055	JSB	EXCH\$UTIL_BLOCK_CHECK	
		52 008200F4	8F D0 00057	MOVL	#8519924, R2	: 0932
		51 0222	8F 3C 0005E	MOVZWL	#546, R1	

EXCH\$RTNAM
V04-000

RT-11 name manipulation routines
rtnam_parse_cleanup

M 9
16-Sep-1984 01:21:55
14-Sep-1984 12:29:08

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCRTNAM.B32;1

Page 29
(11)

		50	65	D0	00063	MOVL	(R5), R0	:	
			67	16	00066	JSB	EXCH\$UTIL_BLOCK_CHECK	:	
14	48	A3	05	E1	00068	BBC	#5, 72(R3), 1\$:	0935
			7E	D4	0006D	CLRL	-(SP)	:	0941
			53	DD	0006F	PUSHL	R3	:	
	00000000G	EF	02	FB	00071	CALLS	#2, EXCH\$RTACP_CONSOLIDATE	:	
			53	DD	00078	PUSHL	R3	:	0945
	00000000G	EF	01	FB	0007A	CALLS	#1, EXCH\$RT11_DIRCACHE_STOP	:	
			65	DD	00081	PUSHL	(R5)	:	0951
	00000000G	EF	01	FB	00083	CALLS	#1, EXCH\$UTIL_RT11CTX_RELEASE	:	
			66	DD	0008A	PUSHL	(R6)	:	0955
	00000000G	EF	01	FB	0008C	CALLS	#1, EXCH\$UTIL_NAMB_RELEASE	:	
			64	DD	00093	PUSHL	(R4)	:	0959
	00000000G	EF	01	FB	00095	CALLS	#1, EXCH\$UTIL_FILB_RELEASE	:	
			04	0009C	RET			:	0962

; Routine Size: 157 bytes, Routine Base: EXCH\$RTNAM_CODE + 06CD

```

: 878 0963 1 GLOBAL ROUTINE rtnam_parse_next_input = %SBTTL 'rtnam_parse_next_input'
: 879 0964 2 BEGIN
: 880 0965 2 ++
: 881 0966 2
: 882 0967 2 FUNCTIONAL DESCRIPTION:
: 883 0968 2
: 884 0969 2 Fetch the next input parameter. Parse the filename and initialize the input file work region.
: 885 0970 2
: 886 0971 2 INPUTS:
: 887 0972 2
: 888 0973 2 none
: 889 0974 2
: 890 0975 2 IMPLICIT INPUTS:
: 891 0976 2
: 892 0977 2 Command qualifier value as returned from CLI$xxx routines. rtnam command work area.
: 893 0978 2
: 894 0979 2 OUTPUTS:
: 895 0980 2
: 896 0981 2 none
: 897 0982 2
: 898 0983 2 IMPLICIT OUTPUTS:
: 899 0984 2
: 900 0985 2 Command work area receives parse info
: 901 0986 2
: 902 0987 2 ROUTINE VALUE:
: 903 0988 2
: 904 0989 2 Success or worst error encountered.
: 905 0990 2
: 906 0991 2 SIDE EFFECTS:
: 907 0992 2
: 908 0993 2 none
: 909 0994 2 --
: 910 0995 2
: 911 0996 2 $dbgtrc_prefix ('rtnam_parse_next_input> ');
: 912 0997 2
: 913 0998 2 LOCAL
: 914 0999 2 ctx : $ref_bblock,
: 915 1000 2 volb : $ref_bblock,
: 916 1001 2 format,
: 917 1002 2 status
: 918 1003 2 ;
: 919 1004 2
: 920 1005 2 BIND
: 921 1006 2 rtnam = exch$a_gbl [excg$a_rtnam_work] : $ref_bblock, ! Pointer to work area
: 922 1007 2 inp_filb = rtnam [rtnam$a_inp_filb] : $ref_bblock, ! Filb for the input
: 923 1008 2 inp_namb = rtnam [rtnam$a_inp_namb] : $ref_bblock ! Namb for the input
: 924 1009 2 ;
: 925 1010 2
: 926 1011 2
: 927 1012 2 $block_check (2, .rtnam, rtnam, 542);
: 928 1013 2
: 929 1014 2 ! Fetch the filename and a pointer to a namb
: 930 1015 2
: 931 1016 2 IF NOT (status = exch$cmd_parse_filespec (%ASCII 'INPUT', rtnam [rtnam$q_input_sticky_name], 0,
: 932 1017 2 rtnam [rtnam$q_input_filename], inp_namb))
: 933 1018 2 THEN
: 934 1019 2 BEGIN
```



```

: 935      1020 3      IF .status NEQ 0
: 936      1021 3      THEN
: 937      1022 3          $exch_signal (exch$_parseerr, 1, rtnam [rtnam$q_input_filename], .status);
: 938      1023 3      RETURN .status;          ! No more files to rtnam, or error in parse
: 939      1024 3      END;
: 940      1025 2      $trace_print_fao ('input parameter is "'AS'", rtnam [rtnam$q_input_filename]);
: 941      1026 2
: 942      1027 2      ! If the input potentially describes multiple files, then set the bit
: 943      1028 2      !
: 944      1029 2      IF .inp_namb [namb$v_more_files] OR .inp_namb [namb$v_wildcard]
: 945      1030 2      THEN
: 946      1031 2          rtnam [rtnam$v_multiple_files] = true;
: 947      1032 2
: 948      1033 2      ! If a foreign device is not mounted, then perform an implied mount
: 949      1034 2      !
: 950      1035 3      IF (.inp_namb [namb$a_assoc_volb] EQL 0)
: 951      1036 2      AND
: 952      1037 4          (BEGIN
: 953      1038 4              BIND
: 954      1039 4                  dev = inp_namb [namb$l_fabdev] : $bblock;
: 955      1040 5                  .dev [dev$v_for] OR (NOT (.dev [dev$v_mnt]))
: 956      1041 3              END)
: 957      1042 2      AND
: 958      1043 4          ((.inp_namb [namb$b_devclass] EQL dc$_disk)
: 959      1044 3          OR
: 960      1045 3          (.inp_namb [namb$b_devclass] EQL dc$_tape))
: 961      1046 3      THEN
: 962      1047 3          BEGIN
: 963      1048 3
: 964      1049 4          IF NOT (status = exch$moun_implied_mount (.inp_namb))
: 965      1050 3          THEN
: 966      1051 4              BEGIN
: 967      1052 4                  exch$util_namb_release (.inp_namb);
: 968      1053 4                  RETURN .status;
: 969      1054 3              END;
: 970      1055 2          END;
: 971      1056 2
: 972      1057 2      volb = .inp_namb [namb$a_assoc_volb];
: 973      1058 2      IF .volb NEQ 0          ! If the volb is there, then the interesting format
: 974      1059 2      THEN
: 975      1060 2          format = .volb [volb$b_vol_format] ! is that of the volb. Otherwise, it
: 976      1061 2      ELSE
: 977      1062 2          format = .inp_namb [namb$b_vol_format];      ! is that of the namb.
: 978      1063 2
: 979      1064 2      CASE .format FROM volb$k_vfmt_lobound TO volb$k_vfmt_hibound OF
: 980      1065 2      SET
: 981      1066 2          [volb$k_vfmt_files11] :
: 982      1067 3              BEGIN
: 983      1068 3                  LOCAL
: 984      1069 3                      errstat;
: 985      1070 3                  IF .inp_namb [namb$v_explicit_node]
: 986      1071 3                  THEN
: 987      1072 4                      BEGIN
: 988      1073 4                          errstat = exch$_noremote;
: 989      1074 4                          $exch_signal (.errstat, 1, rtnam [rtnam$q_input_filename]);
: 990      1075 4                      END
: 991      1076 3                  ELSE
```



```

: 992      1077 4      BEGIN
: 993      1078 4      errstat = exch$opnotperf11;
: 994      1079 4      $exch_signal (.errstat, 1, inp_namb [namb$q_device]);
: 995      1080 4      END;
: 996      1081 4      exch$util_namb_release (.inp_namb);      ! Release the bad namb
: 997      1082 4      RETURN .errstat;
: 998      1083 4      END;
: 999      1084 4
: 1000     1085 4      [volb$k_vfmt_dos11] :
: 1001     1086 4      BEGIN
: 1002     1087 4      LOCAL
: 1003     1088 4      errstat;
: 1004     1089 4      errstat = exch$opnotperdos;
: 1005     1090 4      $exch_signal (.errstat, 1, inp_namb [namb$q_device]);
: 1006     1091 4      exch$util_namb_release (.inp_namb);      ! Release the bad namb
: 1007     1092 4      RETURN .errstat;
: 1008     1093 4      END;
: 1009     1094 4
: 1010     1095 4      [volb$k_vfmt_rt11] :
: 1011     1096 4      ;
: 1012     1097 4      ! These we can do
: 1013     1098 4      !\ [volb$k_vfmt_rtmt] :
: 1014     1099 4      !\ BEGIN
: 1015     1100 4      !\ LOCAL
: 1016     1101 4      !\ errstat;
: 1017     1102 4      !\ errstat = exch$opnotperrtmt;
: 1018     1103 4      !\ $exch_signal (.errstat, 1, inp_namb [namb$q_device]);
: 1019     1104 4      !\ exch$util_namb_release (.inp_namb);      ! Release the bad namb
: 1020     1105 4      !\ RETURN .errstat;
: 1021     1106 4      !\ END;
: 1022     1107 4
: 1023     1108 4      [INRANGE, OUTFRANGE] :
: 1024     1109 4      $logic_check (0, (false), 238);
: 1025     1110 4      TES;
: 1026     1111 4
: 1027     1112 4      $logic_check (3, (.volb NEQ 0), 119);
: 1028     1113 4
: 1029     1114 4      ! Now copy the full name to the default name for proper stickiness
: 1030     1115 4      !
: 1031     1116 4      str$copy_dx (rtnam [rtnam$q_input_sticky_name], inp_namb [namb$q_fullname]);
: 1032     1117 4
: 1033     1118 4      ! Allocate a file block to contain the input information
: 1034     1119 4      !
: 1035     1120 4      inp_filb = exch$util_filb_allocate ();
: 1036     1121 4      exch$copy_namb_to_filb (.inp_namb, .inp_filb); ! Copy from the namb to the filb
: 1037     1122 4
: 1038     1123 4      ! Put an RT-11 context block on the filb
: 1039     1124 4      !
: 1040     1125 4      ctx = exch$util_rt11ctx_allocate (.volb, .inp_filb);
: 1041     1126 4      inp_filb [filb$a_context] = .ctx;
: 1042     1127 4
: 1043     1128 4      RETURN .status;
: 1044     1129 4      END;
```

.PSECT EXCH\$RTNAM_PLIT,NOWRT,2


```
00 00 00 54 55 50 4E 49 00040 P.AAL: .ASCII \INPUT\<0><0><0>
                                010E0005 00048 P.AAK: .LONG 17694725
                                00000000 0004C .ADDRESS P.AAL

                                .EXTRN EXCH$_NOREMOTE, EXCH$_OPNOTPERF11
                                .EXTRN EXCH$_OPNOTPERDOS

                                .PSECT EXCH$RTNAM_CODE,NOWRT,2

                                01FC 00000 .ENTRY RTNAM_PARSE_NEXT_INPUT, Save R2,R3,R4,R5,-
                                R6,R7,R8
                                MOVAB EXCH$UTIL_NAMB_RELEASE, R8
                                MOVAB LIB$SIGNAC, R7
                                ADDL3 #24, EXCH$A_GBL, R0
                                MOVL (R0), R3
                                MOVL #3408114, R2
                                MOVZWL #542, R1
                                MOVL R3, R0
                                JSB EXCH$UTIL_BLOCK_CHECK
                                PUSHAB 40(R3)
                                PUSHL R3
                                CLRL -(SP)
                                PUSHAB 20(R3)
                                PUSHAB P.AAK
                                CALLS #5, EXCH$CMD_PARSE_FILESPEC
                                MOVL R0, STATUS
                                BLBS STATUS, 1$
                                BEQL 6$
                                PUSHR #^M<R3,R6>
                                PUSHL #1
                                PUSHL #EXCH$_PARSEERR
                                CALLS #4, LIB$SIGNAL
                                BRB 6$
                                MOVL 40(R3), R2
                                TSTB 109(R2)
                                BLSS 2$
                                BLBC 108(R2), 3$
                                BISB2 #1, 32(R3)
                                TSTL 116(R2)
                                BNEQ 7$
                                BLBS 107(R2), 4$
                                BBS #3, 106(R2), 7$
                                CMPB 120(R2), #1
                                BEQL 5$
                                CMPB 120(R2), #2
                                BNEQ 7$
                                PUSHL R2
                                CALLS #1, EXCH$MOUN_IMPLIED_MOUNT
                                MOVL R0, STATUS
                                BLBS STATUS, 7$
                                PUSHL R2
                                CALLS #1, EXCH$UTIL_NAMB_RELEASE
                                BRW 18$
                                MOVL 116(R2), VOLB
                                BEQL 8$
                                MOVZBL 88(VOLB), FORMAT

00000000G EF 9E 00002 MOVAB
57 00000000G 00 9E 00009 MOVAB
50 00000000G EF 18 C1 00010 ADDL3
53 003400F2 60 D0 00018 MOVL
52 003400F2 8F D0 0001B MOVL
51 021E 8F 3C 00022 MOVZWL
50 00000000G 53 D0 00027 MOVL
00000000G EF 16 0002A JSB
28 A3 9F 00030 PUSHAB
53 DD 00033 PUSHL
7E D4 00035 CLRL
14 A3 9F 00037 PUSHAB
0000 0000 CF 9F 0003A PUSHAB
00000000G EF 05 FB 0003E CALLS
56 50 D0 00045 MOVL
13 56 E8 00048 BLBS
0048 50 13 0004B BEQL
8F BB 0004D PUSHR
01 DD 00051 PUSHL
00000000G 8F DD 00053 PUSHL
67 04 FB 00059 CALLS
3F 11 0005C BRB
52 28 A3 D0 0005E 1$: MOVL
6D A2 95 00062 TSTB
04 19 00065 BLSS
20 04 6C A2 E9 00067 BLBC
A3 01 88 0006B 2$: BISB2
74 A2 D5 0006F 3$: TSTL
2C 12 00072 BNEQ
23 6A 05 6B A2 E8 00074 BLBS
A2 03 E0 00078 BBS
01 78 A2 91 0007D 4$: CMPB
02 78 A2 91 00083 BEQL
17 12 00087 CMPB
00000000G EF 01 FB 0008B 5$: PUSHL
56 50 D0 00092 CALLS
08 56 E8 00095 MOVL
52 DD 00098 BLBS
68 01 FB 0009A PUSHL
0098 31 0009D 6$: CALLS
55 74 A2 D0 000A0 7$: BRW
06 13 000A4 MOVL
50 58 A5 9A 000A6 BEQL
MOVZBL 88(VOLB), FORMAT
```

0050	03 001D	50 00 0036	7A	04 A2 50 0008	11 9A CF	000AA 000AC 8\$: 000B0 9\$: 000B4 10\$:	BRB MOVZBL CASEL .WORD	9\$ 122(R2), FORMAT FORMAT, #0, #3 11\$-10\$,- 14\$-10\$,- 12\$-10\$,- 17\$-10\$	1062 1064
		7E	EE	8F 01 8F 03 33 06 54 53 13 54 07 54 40 67 68 50	9A DD DD FB 11 E1 DO DD 11 DO 11 DO 9F DD DD 03 52 01 54 04 9F 9F 02 00 50 A3 0C 02 53 55 02 50 56	000BC 11\$: 000C0 000C2 000C8 000CF 000D1 12\$: 000D6 000DD 000DF 000E1 13\$: 000E8 000EA 14\$: 000F1 15\$: 000F4 16\$: 000F6 000F8 000FB 000FD 00100 00103 00104 17\$: 00107 0010A 00111 00118 0011C 00120 00122 00129 0012B 0012D 00134 00138 18\$: 0013B	MOVZBL PUSHL PUSHL CALLS BRB BBC MOVL PUSHL BRB MOVL BRB MOVL PUSHAB PUSHL PUSHL CALLS PUSHL CALLS MOVL RET PUSHAB PUSHAB CALLS CALLS MOVL MOVL PUSHR CALLS PUSHL PUSHL CALLS MOVL MOVL RET	#238, -(SP) #1 #EXCH\$ BADLOGIC #3, LIB\$STOP 17\$ #6, 108(R2), 13\$ #EXCH\$ NOREMOTE, ERRSTAT R3 16\$ #EXCH\$ OPNOTPERF11, ERRSTAT 15\$ #EXCH\$ OPNOTPERDOS, ERRSTAT 64(R2) #1 ERRSTAT #3, LIB\$SIGNAL R2 #1, EXCH\$UTIL_NAMB_RELEASE ERRSTAT, R0 24(R2) 20(R3) #2, STR\$COPY_DX #0, EXCH\$UTIC_FILB_ALLOCATE R0, 36(R3) 36(R3), R3 #*M<R2,R3> #2, EXCH\$COPY_NAMB_TO_FILB R3 VOLB #2, EXCH\$UTIL_RT11CTX_ALLOCATE CTX, 32(R3) STATUS, R0	1109 1070 1073 1074 1078 1079 1089 1090 1091 1092 1116 1120 1121 1125 1126 1128 1129

; Routine Size: 316 bytes, Routine Base: EXCH\$RTNAM_CODE + 076A


```
1046 1130 1 GLOBAL ROUTINE exch$rtnam_rename = %SBTTL 'exch$rtnam_rename'
1047 1131 2 BEGIN
1048 1132 2 ++
1049 1133 2
1050 1134 2 FUNCTIONAL DESCRIPTION:
1051 1135 2
1052 1136 2 Entry routine for the RENAME verb for RT11 only
1053 1137 2
1054 1138 2 INPUTS:
1055 1139 2
1056 1140 2 none
1057 1141 2
1058 1142 2 IMPLICIT INPUTS:
1059 1143 2
1060 1144 2 Command parameters and qualifiers as returned from CLI$ routines. Global environment ref'd by exch$
1061 1145 2
1062 1146 2 OUTPUTS:
1063 1147 2
1064 1148 2 none
1065 1149 2
1066 1150 2 IMPLICIT OUTPUTS:
1067 1151 2
1068 1152 2 none
1069 1153 2
1070 1154 2 ROUTINE VALUE:
1071 1155 2
1072 1156 2 Success or worst error encountered.
1073 1157 2
1074 1158 2 SIDE EFFECTS:
1075 1159 2
1076 1160 2 Files may be renamed.
1077 1161 2 --
1078 1162 2
1079 1163 2 $dbgtrc_prefix ('rtnam_rename> ');
1080 1164 2
1081 1165 2 LOCAL
1082 1166 2 protect,
1083 1167 2 rtnam : $ref_bblock, ! pointer to work area
1084 1168 2 status
1085 1169 2 ;
1086 1170 2
1087 1171 2 ! Allocate and/or initialize the work area
1088 1172 2
1089 1173 2 rtnam_init ();
1090 1174 2
1091 1175 2 ! Get pointers that we need. Have to wait until work area allocated by init call
1092 1176 2
1093 1177 2 rtnam = .exch$a_gbl [excg$a_rtnam_work]; ! Pointer to work area
1094 1178 2 rtnam [rtnam$v_rename_command] = true;
1095 1179 2
1096 1180 2 ! For /PROTECT, we need to know whether it was specified or defaulted
1097 1181 2
1098 1182 2 protect = cli$present (%ASCII 'PROTECT');
1099 1183 2 rtnam [rtnam$v_q_protect] = .protect;
1100 1184 4 rtnam [rtnam$v_q_protect_explicit] = ((.protect EQL cli$present) ! Simply value of low bit
1101 1185 2 OR (.protect EQL cli$negated)); ! Either /PROTECT or /NOPROT
1102 1186 2 ! must be there
```

EXCH\$RTNAM
V04-000

RT-11 name manipulation routines
exch\$rtnam_rename

G 10
16-Sep-1984 01:21:55
14-Sep-1984 12:29:08

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCRTNAM.B32;1

Page 36
(13)

```
: 1103      1187 2 ! Do the rest from the common routine
: 1104      1188 2 !
: 1105      1189 2 status = rtnam_common ();
: 1106      1190 2
: 1107      1191 2 IF .rtnam [rtnam$a_out_namb] NEQ 0
: 1108      1192 2 THEN
: 1109      1193 2     exch$util_namb_release (.rtnam [rtnam$a_out_namb]);
: 1110      1194 2
: 1111      1195 2 RETURN .status;
: 1112      1196 1 END;
```

! Return the namb to the available p

```
                                .PSECT EXCH$RTNAM_PLIT,NOWRT,2
                                .ASCII \PROTECT\<0>
                                .LONG 17694727
                                .ADDRESS P.AAN
                                .EXTRN CLIS_$PRESENT, CLIS_$NEGATED
                                .PSECT EXCH$RTNAM_CODE,NOWRT,2
                                .ENTRY EXCH$RTNAM_RENAME, Save R2,R3,R4
                                CALLS #0, RTNAM_INIT
                                MOVL EXCH$a_GBC, R0
                                MOVL 24(R0), RTNAM
                                BISB2 #4, 32(RTNAM)
                                PUSHAB P.AAN
                                CALLS #1, CLIS$PRESENT
                                INSV PROTECT, #1, #1, 28(RTNAM)
                                CLRL R3
                                CMPL PROTECT, #CLIS_$PRESENT
                                BNEQ 1$
                                INCL R3
                                CLRL R1
                                CMPL PROTECT, #CLIS_$NEGATED
                                BNEQ 2$
                                INCL R1
                                BISB3 R3, R1, R4
                                INSV R4, #2, #1, 28(RTNAM)
                                CALLS #0, RTNAM_COMMON
                                MOVL R0, STATUS
                                TSTL 48(RTNAM)
                                BEQL 3$
                                PUSHL 48(RTNAM)
                                CALLS #1, EXCH$UTIL_NAMB_RELEASE
                                MOVL STATUS, R0
                                RET

                                00 54 43 45 54 4F 52 50 00050 P.AAN:
                                010E0007 00058 P.AAM:
                                00000000' 0005C

                                FD5C CF 00000000G 00 FB 00002
                                50 52 18 A0 D0 00007
                                20 A2 0000' 04 88 00012
                                00000000G 00 0000' CF 9F 00016
                                01 01 50 F0 00021
                                00000000G 8F 53 D4 00027
                                00000000G 8F 50 D1 00029
                                00000000G 8F 02 12 00030
                                54 51 53 D6 00032 1$:
                                01 02 51 D4 00034 1$:
                                F70A CF 50 D1 00036 1$:
                                01 53 51 D6 0003F 2$:
                                00000000G EF 30 A2 D5 00053 2$:
                                50 30 A2 DD 00058 2$:
                                00000000G EF 30 01 FB 0005B 3$:
                                50 53 D0 00062 3$:
                                04 00065
```

; Routine Size: 102 bytes, Routine Base: EXCH\$RTNAM_CODE + 08A6


```
1114 1197 1 GLOBAL ROUTINE rtnam_rename_action (res_len, res_buf) = %SBTTL 'rtnam_rename_action (res_len, res_buf)'
1115 1198 2 BEGIN
1116 1199 2 ++
1117 1200 2
1118 1201 2 FUNCTIONAL DESCRIPTION:
1119 1202 2
1120 1203 2     Secondary action routine for the RENAME verb for RT11 only
1121 1204 2
1122 1205 2 INPUTS:
1123 1206 2
1124 1207 2     res_len - length of result name
1125 1208 2     res_buf - address of result name
1126 1209 2
1127 1210 2 IMPLICIT INPUTS:
1128 1211 2
1129 1212 2     Command parameters and qualifiers as returned from CLIS routines.  Global environment ref'd by exch$
1130 1213 2
1131 1214 2 OUTPUTS:
1132 1215 2
1133 1216 2     none
1134 1217 2
1135 1218 2 IMPLICIT OUTPUTS:
1136 1219 2
1137 1220 2     none
1138 1221 2
1139 1222 2 ROUTINE VALUE:
1140 1223 2
1141 1224 2     Success or worst error encountered.
1142 1225 2
1143 1226 2 SIDE EFFECTS:
1144 1227 2
1145 1228 2     Files may be renamed.
1146 1229 2 --
1147 1230 2
1148 1231 2 $dbgtrc_prefix ('rtnam_rename_action> ');
1149 1232 2
1150 1233 2 LOCAL
1151 1234 2     ctx2 : $ref_bblock,           ! context block to look for duplicates
1152 1235 2     rfp : $bblock [nam$c_bln+nam$c_maxrss], ! related file parse - an RMS NAM block plus expanded string
1153 1236 2     fin_buf : $bvector [filb$s_result_name], ! a buffer in which to build the actual final name
1154 1237 2     fin_len,
1155 1238 2     nam_len,
1156 1239 2     status
1157 1240 2 ;
1158 1241 2
1159 1242 2 BIND
1160 1243 2     rtnam = exch$a_gbl [excg$a_rtnam_work] : $ref_bblock,           ! Pointer to work area
1161 1244 2     out_name = rtnam [rtnam$a_output_filename] : $desc_block,
1162 1245 2     nam$b = rtnam [rtnam$a_out_nam$b] : $ref_bblock,
1163 1246 2     filb = rtnam [rtnam$a_inp_filb] : $ref_bblock,
1164 1247 2     volb = filb [filb$a_assoc_volb] : $ref_bblock,
1165 1248 2     ctx = filb [filb$a_context] : $ref_bblock,
1166 1249 2     ent = ctx [rt11ctx$a_ent_address] : $ref_bblock
1167 1250 2 ;
1168 1251 2
1169 1252 2 $block_check (2, .filb, filb, 450);
1170 1253 2 $block_check (2, .volb, volb, 460);
```

```
1171 1254 2 $block_check (2, .ctx, rt11ctx, 444);
1172 1255 2
1173 1256 2 ! If we haven't fetched the output name, do so now. This gives us a chance to carry the sticky device over
1174 1257 2 output name
1175 1258 2
1176 1259 2 IF NOT .rtnam [rtnam$sv_out_fetched]
1177 1260 2 THEN
1178 1261 2 BEGIN
1179 1262 2 LOCAL
1180 1263 2 desc : VECTOR [2, LONG];
1181 1264 2
1182 1265 2 desc [0] = .res_len;
1183 1266 2 desc [1] = .res_buf;
1184 1267 2
1185 1268 2 ! Fetch the desired filename and a pointer to a namb
1186 1269 2
1187 1270 2 IF NOT (status = exch$cmd_parse_filespec (%ASCII 'OUTPUT', desc, 0, rtnam [rtnam$q_output_filename], nam
1188 1271 2 THEN
1189 1272 2 $exch_signal_return (exch$parseerr, 1, rtnam [rtnam$q_output_filename], .status);
1190 1273 2 rtnam [rtnam$a_out_namb] = .namb; ! Save the namb pointer
1191 1274 2 $trace_print_fao ("output '!AS'", expanded "'!AS'", rtnam [rtnam$q_output_filename], namb [namb$q_fullnam
1192 1275 2
1193 1276 2 ! If we have been given an invalid filename, then choke
1194 1277 2
1195 1278 2 IF .namb [namb$sv_bad_pdp_char]
1196 1279 2 OR
1197 1280 2 .namb [namb$sv_rt_truncate]
1198 1281 2 THEN
1199 1282 2 $exch_signal_return (exch$_badfilename, 3, out_name, .volb [volb$l_vol_type_len], volb [volb$t_vol_t
1200 1283 2
1201 1284 2 rtnam [rtnam$sv_out_fetched] = true;
1202 1285 2
1203 1286 2 END;
1204 1287 2
1205 1288 2 ! Perform a related file parse to get the final name
1206 1289 2
1207 1290 2 IF NOT (status = exch$cmd_related_file_parse (
1208 1291 2 .out_name [dsc$w_length], .out_name [dsc$a_pointer], ! Command line name string
1209 1292 2 .res_len, .res_buf, ! Related file name
1210 1293 2 rfp))
1211 1294 2 THEN
1212 1295 2 $exch_signal_return (exch$_badfilename, 3, out_name, .volb [volb$l_vol_type_len], volb [volb$t_vol_type]
1213 1296 2
1214 1297 2 nam_len = .rfp [nam$b_name] + .rfp [nam$b_type];
1215 1298 2 fin_len = .volb [volb$l_vol_ident_len] + .nam_len; ! Length of volume ident plus file name
1216 1299 2 $logic_check (2, (.fin_len [EQU filb$s_result_name], 161);
1217 1300 2 CH$COPY (.volb [volb$l_vol_ident_len], volb [volb$t_vol_ident], ! Volume name
1218 1301 2 .nam_len, .rfp [nam$l_name], 0, filb$s_result_name, fin_buf);
1219 1302 2
1220 1303 2 $trace_print_fao ('related '!AF', final '!AF'', .res_len, .res_buf, .fin_len, fin_buf);
1221 1304 2
1222 1305 2 ! Make sure he isn't trying to move it between devices
1223 1306 2
1224 1307 2 IF .volb NEQ .namb [namb$a_assoc_volb]
1225 1308 2 THEN
1226 1309 2 $exch_signal_return (exch$_norendev);
1227 1310 2
```



```
: 1228      1311 2 ! We need to make sure that the output name doesn't already exist
: 1229      1312 2 !
: 1230      1313 2 ctx2 = exch$util_rt11ctx_allocate (.volb, 0); ! Get a context block so we can look
: 1231      1314 2 IF exch$rtacp_find_file (.ctx2, .rfp [nam$l_name], .nam_len)
: 1232      1315 2 AND
: 1233      1316 2 .ent NEQA .ctx2 [rt11ctx$a_ent_address] ! Let him rename it to the same, necessary for RENAME /PROTE
: 1234      1317 2 THEN
: 1235      1318 2 $exch_signal (exch$_norenexists, 4, .res_len, .res_buf, .fin_len, fin_buf)
: 1236      1319 2
: 1237      1320 2 ! Everything is ok, do the rename
: 1238      1321 2
: 1239      1322 2 ELSE
: 1240      1323 2 BEGIN
: 1241      1324 2
: 1242      1325 2 ! Convert the name to radix50 and put in into the directory buffer
: 1243      1326 2
: 1244      1327 2 exch$util_radix50_from_ascii (.rfp [nam$b_name], .rfp [nam$l_name],
: 1245      1328 2 rt11ctx$s_exp_name, ent [rt11ent$l_filename]);
: 1246      1329 2 exch$util_radix50_from_ascii (.rfp [nam$b_type] - 1, .rfp [nam$l_type] + 1,
: 1247      1330 2 rt11ctx$s_exp_type, ent [rt11ent$w_filetype]);
: 1248      1331 2
: 1249      1332 2 ! Also change the name in the context buffer, since it will be used by EXCH$RTACP_CHECK_POSITION
: 1250      1333 2
: 1251      1334 2 ctx [rt11ctx$l_filename] = .ent [rt11ent$l_filename];
: 1252      1335 2 ctx [rt11ctx$w_filetype] = .ent [rt11ent$w_filetype];
: 1253      1336 2
: 1254      1337 2 ! If a protection change was requested, do it here
: 1255      1338 2
: 1256      1339 2 IF .rtnam [rtnam$v_q_protect_explicit]
: 1257      1340 2 THEN
: 1258      1341 2 ent [rt11ent$v_typ_protected] = ctx [rt11ctx$v_typ_protected] = .rtnam [rtnam$v_q_protect];
: 1259      1342 2
: 1260      1343 2 ! Update the directory
: 1261      1344 2
: 1262      1345 2 exch$rt11_dirseg_put (.volb, .ctx [rt11ctx$l_seg_number]); ! Write the directory segment
: 1263      1346 2
: 1264      1347 2 ! Log the action if requested
: 1265      1348 2
: 1266      1349 2 IF .rtnam [rtnam$v_q_log]
: 1267      1350 2 THEN
: 1268      1351 2 $exch_signal (exch$_renamed, 4, .res_len, .res_buf, .fin_len, fin_buf);
: 1269      1352 2 END;
: 1270      1353 2
: 1271      1354 2 ! Release the context block
: 1272      1355 2
: 1273      1356 2 exch$util_rt11ctx_release (.ctx2);
: 1274      1357 2
: 1275      1358 2 RETURN true;
: 1276      1359 1 END;
```

.PSECT EXCH\$RTNAM_PLIT,NOWRT,2

```
00 00 54 55 50 54 55 4F 00060 P.AAP: .ASCII \OUTPUT\<0><0>
010E0006 00068 P.AAO: .LONG 17694726
00000000 0006C .ADDRESS P.AAP
```

				OFFC	00000			ENTRY	RTNAM_RENAME_ACTION, Save R2,R3,R4,R5,R6,-	
50	00000000G	5E	FD90	CE	9E	00002		MOVAB	-624(SP), SP	1197
		EF		18	C1	00007		ADDL3	#24, EXCH\$A_GBL, R0	1243
		56		60	DO	0000F		MOVL	(R0), R6	1244
		53	OC	A6	9E	00012		MOVAB	12(R6), R3	
54	24	A6		1C	C1	00016		ADDL3	#28, 36(R6), R4	1247
50	24	A6		20	C1	0001B		ADDL3	#32, 36(R6), R0	1248
		58		60	DO	00020		MOVL	(R0), R8	1249
		52	035B00FA	8F	DO	00023		MOVL	#56295674, R2	1252
		51	01C2	8F	3C	0002A		MOVZWL	#450, R1	
		50	24	A6	DO	0002F		MOVL	36(R6), R0	
			00000000G	EF	16	00033		JSB	EXCH\$UTIL_BLOCK_CHECK	
		57		64	DO	00039		MOVL	(R4), R7	1253
		52	041B00F3	8F	DO	0003C		MOVL	#68878579, R2	
		51	01CC	8F	3C	00043		MOVZWL	#460, R1	
		50		57	DO	00048		MOVL	R7, R0	
			00000000G	EF	16	0004B		JSB	EXCH\$UTIL_BLOCK_CHECK	
		52	008200F4	8F	DO	00051		MOVL	#8519924, R2	1254
		51	01BC	8F	3C	00058		MOVZWL	#444, R1	
		50		58	DO	0005D		MOVL	R8, R0	
			00000000G	EF	16	00060		JSB	EXCH\$UTIL_BLOCK_CHECK	
47	20	A6		03	EO	00066		BBS	#3, 32(R6), 2\$	1259
	08	AE	04	AC	7D	0006B		MOVQ	RES_LEN, DESC	1265
			30	A6	9F	00070		PUSHAB	48(R6)	1270
				53	DD	00073		PUSHL	R3	
				7E	D4	00075		CLRL	-(SP)	
			14	AE	9F	00077		PUSHAB	DESC	
			0000'	CF	9F	0007A		PUSHAB	P.AAO	
00000000G		EF		05	FB	0007E		CALLS	#5, EXCH\$CMD_PARSE_FILESPEC	
		54		50	DO	00085		MOVL	R0, STATUS	
		16		54	E8	00088		BLBS	STATUS, 1\$	
		52	00000000G	8F	DO	0008B		MOVL	#EXCH\$_PARSEERR, TEMP	1272
				18	BB	00092		PUSHR	#^M<R3,R4>	
				01	DD	00094		PUSHL	#1	
				52	DD	00096		PUSHL	TEMP	
00000000G	00			04	FB	00098		CALLS	#4, LIB\$SIGNAL	
				46	11	0009F		BRB	4\$	
		50	30	A6	DO	000A1	1\$:	MOVL	48(R6), R0	1273
		24	6E	A0	E8	000A5		BLBS	110(R0), 3\$	1278
1F	6E	A0		02	EO	000A9		BBS	#2, 110(R0), 3\$	1280
	20	A6		08	88	000AE		BISB2	#8, 32(R6)	1284
			0110	CE	9F	000B2	2\$:	PUSHAB	RFP	1290
		7E	04	AC	7D	000B6		MOVQ	RES_LEN, -(SP)	1292
			04	A3	DD	000BA		PUSHL	4(R3)	1291
		7E		63	3C	000BD		MOVZWL</		

			59	A7	DD	000D7	PUSHL	89(R7)		
				53	DD	000DA	PUSHL	R3		
				03	DD	000DC	PUSHL	#3		
				52	DD	000DE	PUSHL	TEMP		
		00000000G	00	05	FB	000E0	CALLS	#5, LIB\$SIGNAL		
				71	11	000E7	BRB	8\$		
			50	FEDB	CD	9A 000E9	MOVZBL	RFP+59, R0	1297	
			51	FEDC	CD	9A 000EE	MOVZBL	RFP+60, R1		
04	6E		50		51	C1 000F3	ADDL3	R1, R0, NAM_LEN	1298	
AE		65	A7		6E	C1 000F7	ADDL3	NAM_LEN, 10T(R7), FIN_LEN	1299	
		00000100	8F	04	AE	D1 000FD	CMPL	FIN_LEN, #256		
					13	1B 00105	BLEQU	6\$		
			7E	A1	8F	9A 00107	MOVZBL	#161, -(SP)		
					01	DD 0010B	PUSHL	#1		
					8F	DD 0010D	PUSHL	#EXCH\$ BADLOGIC		
		00000000G	00		03	FB 00113	CALLS	#3, LIB\$STOP		
			5B	FEEC	CD	D0 0011A	MOVL	RFP+76, R11	1301	
			5A	0100	8F	3C 0011F	MOVZWL	#256, R10	1300	
			59	10	AE	9E 00124	MOVAB	FIN_BUF, R9		
5A	00	69	A7	65	A7	2C 00128	MOVCS	101(R7), 105(R7), #0, R10, (R9)		
					69	0012F				
					0E	18 00130	BGEQ	7\$		
			59	65	A7	C0 00132	ADDL2	101(R7), R9		
			5A	65	A7	C2 00136	SUBL2	101(R7), R10		
5A	00		6B		6E	2C 0013A	MOVCS	NAM_LEN, (R11), #0, R10, (R9)		
					69	0013F				
			50	30	A6	D0 00140	MOVL	48(R6), R0	1307	
		74	A0		57	D1 00144	CMPL	R7, 116(R0)		
					14	13 00148	BEQL	9\$		
			52	00000000G	8F	D0 0014A	MOVL	#EXCH\$_NORENDEV, TEMP	1309	
					52	DD 00151	PUSHL	TEMP		
		00000000G	00		01	FB 00153	CALLS	#1, LIB\$SIGNAL		
			50		52	D0 0015A	MOVL	TEMP, R0		
					04	0015D	RET			
					7E	D4 0015E	CLRL	-(SP)	1313	
					57	DD 00160	PUSHL	R7		
		00000000G	EF		02	FB 00162	CALLS	#2, EXCH\$UTIL_RT11CTX_ALLOCATE		
			53		50	D0 00169	MOVL	R0, CTX2		
					6E	DD 0016C	PUSHL	NAM_LEN	1314	
				0808	8F	BB 0016E	PUSHR	#^MZR3,R11>		
		00000000G	EF		03	FB 00172	CALLS	#3, EXCH\$RTACP_FIND_FILE		
			1B		50	E9 00179	BLBC	R0, 10\$		
			7E	A3	7E	A8	D1 0017C	CMPL	126(R8), 126(CTX2)	1316
					14	13 00181	BEQL	10\$		
				10	AE	9F 00183	PUSHAB	FIN_BUF	1318	
				08	AE	DD 00186	PUSHL	FIN_LEN		
			7E	04	AC	7D 00189	MOVQ	RES_LEN, -(SP)		
					04	DD 0018D	PUSHL	#4		
				00000000G	8F	DD 0018F	PUSHL	#EXCH\$_NORENEXISTS		
					73	11 00195	BRB	12\$		
			52	7E	A8	D0 00197	MOVL	126(R8), R2	1328	
				02	A2	9F 0019B	PUSHAB	2(R2)		
					06	DD 0019E	PUSHL	#6		
					5B	DD 001A0	PUSHL	R11		
			7E	FEDB	CD	9A 001A2	MOVZBL	RFP+59, -(SP)		
		00000000G	EF		04	FB 001A7	CALLS	#4, EXCH\$UTIL_RADIX50_FROM_ASCII		
				06	A2	9F 001AE	PUSHAB	6(R2)	1330	

EXCH\$RTNAM
V04-000

RT-11 name manipulation routines
rtnam_rename_action (res_len, res_buf)

M 10
16-Sep-1984 01:21:55
14-Sep-1984 12:29:08

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCRTNAM.B32;1

Page 42
(14)

39	50	1C	7E	FEFO	CD	03	DD	001B1	PUSHL	#3	:	
01	A8				7E	01	C1	001B3	ADDL3	#1, RFP+80, -(SP)	:	1329
	A2					CD	9A	001B9	MOVZBL	RFP+60, -(SP)	:	
						6E	D7	001BE	DECL	(SP)	:	
						04	FB	001C0	CALLS	#4, EXCH\$UTIL_RADIX50_FROM_ASCII	:	1330
						A2	D0	001C7	MOVL	2(R2), 58(R8)	:	1334
						A2	B0	001CC	MOVW	6(R2), 62(R8)	:	1335
						02	E1	001D1	BBC	#2, 28(R6), 11\$:	1339
						01	EF	001D6	EXTZV	#1, #1, 28(R6), R0	:	1341
						50	F0	001DC	INSV	R0, #7, #1, 57(R8)	:	
						50	F0	001E2	INSV	R0, #7, #1, 1(R2)	:	
						A8	DD	001E8	PUSHL	118(R8)	:	1345
						57	DD	001EB	PUSHL	R7	:	
						02	FB	001ED	CALLS	#2, EXCH\$RT11_DIRSEG_PUT	:	
						A6	E9	001F4	BLBC	28(R6), 13\$:	1349
						AE	9F	001F8	PUSHAB	FIN_BUF	:	1351
						AE	DD	001FB	PUSHL	FIN_LEN	:	
						AC	7D	001FE	MOVQ	RES_LEN, -(SP)	:	
						04	DD	00202	PUSHL	#4	:	
						8F	DD	00204	PUSHL	#EXCH\$ RENAMED	:	
						06	FB	0020A	CALLS	#6, LIB\$SIGNAL	:	
						53	DD	00211	PUSHL	CTX2	:	1356
						01	FB	00213	CALLS	#1, EXCH\$UTIL_RT11CTX_RELEASE	:	
						01	D0	0021A	MOVL	#1, R0	:	1358
						04	0021D		RET		:	1359

; Routine Size: 542 bytes, Routine Base: EXCH\$RTNAM_CODE + 090C

EXCH\$RTNAM
V04-000

RT-11 name manipulation routines
rtnam_rename_action (res_len, res_buf)

N 10
16-Sep-1984 01:21:55
14-Sep-1984 12:29:08

VAX-11 Bliss-32 V4.0-742
[EXCHNG.SRC]EXCRTNAM.B32;1

Page 43
(15)

: 1278
: 1279
1360 1 END
1361 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

Name	Bytes	Attributes
EXCH\$RTNAM_PLIT	112 NOVEC,NOWRT, RD	EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)
EXCH\$RTNAM_CODE	2858 NOVEC,NOWRT, RD	EXE,NOSHR, LCL, REL, CON,NOPI,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	17	0	1000	00:01.9
_\$255\$DUA28:[EXCHNG.OBJ]EXCLIB.L32;1	1151	139	12	79	00:01.3

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS\$:EXCRTNAM/OBJ=OBJ\$:EXCRTNAM MSRC\$:EXCRTNAM/UPDATE=(ENHS\$:EXCRTNAM)

: Size: 2858 code + 112 data bytes
: Run Time: 00:49.1
: Elapsed Time: 02:32.2
: Lines/CPU Min: 1662
: Lexemes/CPU-Min: 19137
: Memory Used: 217 pages
: Compilation Complete

